Investigating optical flow and tracking techniques for recovering motion within image sequences

Etienne Corvee

A thesis submitted in fulfilment of the requirements
of Kingston University for the degree of Doctor of Philosophy

Septembre 2005

Abstract

Analysing objects interacting in a 3D environment and captured by a video camera requires knowledge of their motions. *Motion estimation* provides such information, and consists of recovering 2D image velocity, or *optical flow*, of the corresponding moving 3D objects. A *gradient-based* optical flow estimator is implemented in this thesis to produce a dense field of velocity vectors across an image. An iterative and parameterised approach is adopted which fits *planar motion models* locally on the image plane. Motion is then estimated using a *least-squares* minimisation approach. The possible approximations of the optical flow derivative are shown to differ greatly when the magnitude of the motion increases. However, the widely used derivative term remains the optimal approximation to use in the range of accuracies of the gradient-based estimators *i.e.* small motion magnitudes.

Gradient-based estimators do not estimate motion robustly when noise, large motions and multiple motions are present across object boundaries. A *robust statistical* and multi-resolution estimator is developed in this study to address these limitations. Despite significant improvement in performance, the multiple motion problem remains a major limitation. A confidence measurement is designed around optical flow covariance to represent motion accuracy, and is shown to visually represent the lack of robustness across motion boundaries.

The recent *hyperplane* technique is also studied as a global motion estimator but proved unreliable compared to the gradient-based approach. A computationally expensive optical flow estimator is then designed for the purpose of detecting at frame-rate moving objects occluding background scenes which are composed of static objects captured by moving pan and tilt cameras. This was achieved by adapting the estimator to perform *global motion estimation i.e.* estimating the motion of the background scenes. Moving objects are segmented from a thresholding operation on the grey-level differences between motion compensated background frames and captured frames. Filtering operations on small object dimensions and using moving edge information produced reliable results with small levels of noise. The issue of tracking moving objects is studied with the specific problem of data correspondence in occlusion scenarios.

Acknowledgements

I would like to give grateful thanks to

- my supervisor Dr. Graeme A. Jones for his expertise, his continuous guidance, his valuable advice and patience during writing-up and without whom this thesis would have never been completed.
- Dr. Sergio Velastin for giving me the opportunity to work as a research assistant for the European project PRISMATICA[1] which allowed me to meet and work with various European partners. I would also like to thank him for his trust throughout this challenging research.
- Sarah Barman and Anusha Toniappa for the opportunity to work on the design of algorithms for medical applications involving retinal images.
- the research vision laboratory, in particular John-Paul Renno, Anusha Toniappa, David Thirde and Jonathan Rymel for their help and friendship. I would also like to thank Dr. Paul Giaccone, Dr. Darrel Greenhill and Dr. Andreas Hoppe for their assistance; as well as many researchers from the chemistry department for their valuable friendship. Last but not least, I would like to thank my family for their unique encouragement and help.

[1] http://www.prismatica.com/

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The general motivation of this study, discussed in the first section, is the investigation of grey-level changes in image sequences caused essentially by the motion of 3D objects projected onto image plane of cameras for motion detection purposes in surveillance applications. The last section introduces the organisation of the thesis and the content of the chapters.

1.1 Motivation

Rays of light reflected by a scene and captured by a camera produce grey-level patterns on the image plane of the camera. The apparent instantaneous shift of grey-level structures in the image plane due to relative motion between scene objects and camera is referred to as *optical flow* [68]. However, modern cameras capture images of a scene with relatively small time intervals so that the grey-level variations due to the change of scene lighting environment are neglected. Analysing such grey-level motion is referred to as motion analysis and is mainly performed by two approaches. The first approach estimates motion of features, or structures, and produces sparse motion vectors across a frame. The second approach produces dense optical flow (*i.e.* at each pixel in a frame) by using the constant brightness approach approach. Gradient-based motion estimators are one family of optical flow estimator to produce such dense motion information by analysing grey-level gradients. A gradient-based technique is developed and investigated in this study whose estimation is parameterised by a motion model of the projected motion of 3D object surfaces.

Knowing pixel motion between successive images is a powerful tool for computer vision applications as described in the review in chapter 2. However, the use of dense optical flow vectors is limited by the accuracy of the results at object boundaries where multiple motions occur and cannot be easily segmented. In addition, producing dense motion fields is a very computationally expensive task. The second motivation of this study is how to use optical flow in frame-rate applications and the one chosen here concerns detecting moving objects surveyed in indoor and outdoor scenes captured by either static or moving PTZ (pan, tilt and zoom)

cameras involved. Detecting moving objects in such scenarios is usually performed by *pixel differencing* which consists of thresholding grey-level differences. When background scenes, assumed to cover the majority of the frame dimensional space, are non-moving relative to a camera (with no zooming operation involved) moving pixels are easily segmented from the background. Grouping moving pixels into individual objects, referred to as *blobs* allow temporal analysis of their motions on the image plane and they can be therefore tracked using trajectory models. However, although such traditional procedure is efficient, tracking becomes a significant challenge when the problem of occlusion and fragmentation occur where the objects shape and their grey-level distribution do not match between frames. These problems are here addressed by using *appearance models* of the objects features to improve tracking. Moreover, an additional problem occurs when PTZ cameras are allowed to move where the pixels of the background scene move with the camera motions. The dense optical flow algorithm previously introduced here is designed to estimate the global motion of the background scene before pixel differencing is possible and to allow the subsequent tracking of moving objects in frame-rate PTZ applications. This global motion algorithm could permit, for example, the detection of a moving object before being tracked and recognised automatically by automatic adjustments of the pan, tilt and zoom of a camera.

1.2 Structure

The second chapter reviews the main approaches used in the literature to estimate optical flow. Much of this review focuses on the gradient-based approach. A gradient-based technique is then developed in chapter 3 around the constant brightness assumption. In this chapter, a planar motion model is fitted to local neighbourhoods of pixels representing the projected 3D object surfaces on the image plane. Chapter 4 investigates the main limitations of the optical flow estimator caused by the contamination of pixel neighbourhoods by noise and multiple motions and caused by the small motion restriction problem. The problem of estimating the global motion of background scenes captured by moving pan and tilt cameras is addressed by a hyperplane estimator in chapter 5. This technique does not compete with the version of the gradient-based estimator developed in chapter 3 which can estimate background motions using a designed edge-based sampling technique described in chapter 6. In this chapter, the gradient-based estimator is tested in two applications, first a pan and tilt application and second a zooming application. When background scenes are not moving relative to a camera, foreground moving objects are accurately segmented in chapter 7. In this chapter, the problem of tracking these objects in occlusion scenarios is addressed. This study concludes in chapter 8 with a successful segmentation of moving objects in static background scenes captured using moving pan and tilt cameras. Chapter 9 reviews the investigations achieved in this study on dense motion analysis and on the results obtained with the surveillance applications.

Chapter 2

Review

Optical flow is the prominent source of temporal variations in image sequences. The relative movement between a 3D scene and the camera induces apparent motion in image sequences as explained, for example, by Stiller *et al* [123]. One of the key problems in dynamic image analysis arises from the fact that motion is geometric in nature but manifests itself as a change of image pixel intensity. Therefore, the core of the image motion estimation problem concerns relating time-varying image intensity to the movement of objects in the scene. Motion analysis in computer vision is divided in two main areas of study: motion estimation and motion segmentation.

Motion estimation aims to study the object motions that gives rise to the pixel changes, and is based on the assumption that object structures remain relatively rigid from frame to frame. Such analysis is usually based on a further assumption that pixel intensity generated by the 3D scene remains relatively constant over short period of times [1, 4, 10, 15, 19, 63, 68, 97, 123, 141].

The goal of motion segmentation is to partition the image into regions that have uniform motion characteristics or properties [21, 69, 95, 133, 117, 138]. Motion boundaries generally coincide with pixel intensity boundaries (though the converse is not true). Therefore, intensity boundaries can be used to hypothesise motion boundaries.

2.1 Motion estimation

Two distinct approaches have been developed for the computation of motion from image sequences. The first of these is based on extracting a set of relatively sparse but highly discriminatory 2D features in the image plane. These features correspond to the perspective projection of 3D object features such as edges as in the works of Bouthemy [20], corners as studied by Jones *et al* [74], occluding boundaries of surfaces, boundaries demarcating changes in surface reflectivity, *etc.* Such features in the image plane can be modelled as lines, texture information, curves and other possible geometrical shapes. In order to match features from one frame to another in the image sequence, constraints are formulated based on many assumptions such

as rigid body motion *i.e.* that the 3D distance between two features on a rigid body remains the same after both object and camera motion. Matching techniques using this first approach to compute motion from features exploit what may be called the *constant structure constraint.* The result is a sparse set of motion vectors in each frame. Several techniques exist to extract and establish feature correspondence but in general the task is difficult, mainly because of the presence of occlusion, which may cause features to be hidden, false features to be generated and hidden features to reappear as investigated by Girard *et al* [45] and Baumberg *et al* [14]. The second approach to motion estimation is based on computing the *optical flow* *i.e.* the visual velocities of each pixel in the image plane [15]. Optical flow is the apparent instantaneous shift of grey-level structures in the image plane due to relative motion between scene objects and camera (as defined by Horn [68]). This approach to motion estimation is more appropriate to applications which require dense motion fields *i.e.* every pixel in the image should have a motion vector associated with it. In order to generate dense optical flow fields, the *constant brightness equation* may be used, which states that the intensity of a pixel undergoing motion should remain relatively constant from frame to frame. Two major matching techniques for computing optical flow are derived from the constant brightness equation: the *correlation* (or *block-matching*) technique and *gradient-based* technique.

Two comprehensive papers on the subject of optical flow performance exist. Barron *et al* [13] compared nine state of the art optical flow algorithms on the basis of accuracy and density. The authors provide a clear test set of image sequences that can be used for quantitative and qualitative comparison of the different algorithms. More recent work by Liu *et al* [87] has improved on Barron *et al* study. Performing 2D motion detection involves the processing of scenes where the sensor is moving within an environment containing both stationary and moving objects. Furthermore, visual events such as occlusion, transparent motions and non-rigid objects increase the inherent complexity of the measurement of optical flow.

Having estimated a reliable 2D image motion, optical flow may then be used to recover the 3D motion of the visual sensor (to within a scale factor) and the 3D surface structure (shape or relative depth). This is achievable through assumptions concerning the structure of the optical flow field and the type of motion exhibited by the sensor as studied by Adiv [1]. Optical flow may also be used to perform motion detection [70], object segmentation and tracking [8, 135], motion-compensation [41, 43], motion analysis of oceanographic and atmospheric image sequences [37] or stereo disparity measurement [15, 125].

2.2 Correlation-based methods

Window-matching, block-matching or correlation-based techniques are the most intuitive and one of the most widely applied techniques to compute optical flow from an image sequence as

performed by Giachetti *et al* [59] and Sun *et al* [124]. These correlation-based methods analyse the grey-level pattern around the point of interest and search the most similar pattern in the successive image. The search of best match is performed within a chosen search window between a certain numbers of frames, typically neighbouring pairs. The estimated image displacement (or optical flow) is taken as the shift corresponding to the minimum of a distance metric, or maximum of a correlation measure between the different intensity patterns of the different frames. Motion estimation between two patterns is typically achieved by a normalised cross-correlation function such as the sum-of-squared difference (SSD). This technique is also referred to as a least-squares minimisation technique.

Generally, features such ascorner points are successfully matched. However, such points do not constitute the majority of pixels within an image and denser features such as edges are easily mismatched. Moreover, establishing the correct correspondences of reasonably unique features such as the corner points can be problematic. Bouthemy [20] shows for example that edges that can be modelled as 3D step edges can be successfully tracked.

Phase-based techniques are another method to estimate motion [32, 36, 57]. They analyse the image grey-level motion in the frequency space using discrete Fourier analysis and use the property that motion in the spatial domain is equivalent to a convolution in the frequency domain.

For region-based techniques referred to as the block-matching techniques, motion is measured by analysing displaced grey-level differences. The larger the template window, the more efficient the match as there are more pixel comparisons. However, time computation increases quadratically when increasing the template window which is likely to contain pixels belonging to other objects undergoing different motion. This occurs when pixels are closed to object boundaries. Moreover, motions can be expected to be large in magnitude, requiring a large search window to ensure the match is found (where there is no prior information provided on the pixel motion). Such large search windows increase the potential for false matches particularly in the presence of noise, blur or repeated structures. Therefore a trade-off exists in choosing the size of the template and search window which can be estimated if information about the objects to match are known, such as their sizes and their velocities. For large motions, initialisation techniques (reviewed in section 2.8) can limit the search area.

2.3 Gradient-based methods

The image brightness of the projection of a single point is assumed to remain constant with time. This is strictly true only in the idealised context of Lambertian[1] surfaces being viewed by

[1]Dull, matte surfaces can be modelled by Lambertian surfaces and are said to exhibit Lambertian reflection, or diffuse reflection (Nagel [97]). The assumption is that these surfaces reflect light with equal intensity in all directions, and hence appear equally bright from all directions. For a given surface, the brightness depends only

a moving camera. This is a reasonable approximation for a wide range of practical situations. Let $I(x,y,t)$ be the irradiance at time t of the image point located at (x,y) in pixel unit on the vertical and horizontal axis respectively. If (u,v) is the grey-level velocity at location (x,y) then if no occlusion occurs the irradiance remains unchanged at this location within a small interval of time δt, or from time t to $t+\delta t$:

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t) \tag{2.1}$$

where

$$\delta x = u.\delta t, \text{ and } \delta y = v.\delta t \tag{2.2}$$

Equation 2.1 is known as the constant brightness equation (CBE). However, the 2 unknowns of the optical flow displacement, $(\delta x, \delta y)$, cannot be directly estimated by the single constraint of equation 2.1. Using the fact that δt is small so that any 3D object motion induces image optical flow with relatively small displacement allows the second term of the CBE to be expanded into its Taylor series:

$$\begin{aligned} I(x+\delta x, y+\delta y, t+\delta t) &\approx I(x,y,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \epsilon & (2.3)\\ &\approx I(x,y,t) + I_x\delta x + I_y\delta y + I_t\delta t + \epsilon & (2.4) \end{aligned}$$

where,

$$I_x = \frac{\partial I}{\partial x},\ I_y = \frac{\partial I}{\partial y} \text{ and } I_t = \frac{\partial I}{\partial t} \tag{2.5}$$

and ϵ contains the remaining higher order terms in δx, δy, and δt, and $I_x = \partial I/\partial x$, $I_y = \partial I/\partial y$ and $I_t = \partial I/\partial t$. Using equation 2.4 in the CBE of equation 2.1 gives

$$\begin{aligned} I(x,y,t) + I_x\delta x + I_y\delta y + I_t\delta t + \epsilon &\approx I(x,y,t) & (2.6)\\ I_x\delta x + I_y\delta y + I_t\delta t + \epsilon &\approx 0 & (2.7)\\ I_x u + I_y v + I_t &\approx -\epsilon/\delta t & (2.8) \end{aligned}$$

The right hand side of equation 2.8 represents the remaining terms of the Taylor expansion. This contains products of higher spatial and temporal derivatives of the brightness function as well as higher orders of the displacements. Such derivatives are estimated from the image using various kernel filters as evaluated by Christmas [34]. The validity of ignoring the right hand side of equation 2.8 is dependent on the spatial frequency content of the local intensity pattern and the magnitude of the displacement (Chin [33]). However, the remaining terms of the Taylor expansion can in general be neglected which simplifies equation 2.8 to

$$I_x u + I_y v + I_t \approx 0 \tag{2.9}$$

The CBE equation 2.9 is also referred to as the *optical flow constraint* equation and is the equation to be solved in gradient-based optical flow methods. Unlike equation 2.1, the two

on the angle between the direction of the light source and the surface normal

motion unknowns can be estimated if additional constraints are provided, as described in the following sections. In practice, discretisation of the image sequence in time and space affects the equivalence between the intensity invariance assumption and hence affects the validity of equation 2.9. Two ways to satisfy the CBE are to shorten the temporal sampling rate or to somehow reduce the high frequency components in the intensity function. The latter can be achieved by pre-smoothing or intentionally blurring the images before gradients are computed as achieved by Kearney [80] in order to diminish the second and higher order brightness gradients. Pre-smoothing also reduces the effect of noise in the brightness measurement by providing spatial averaging. The optical flow constraint equation holds in high temporal sampling in distant imagery by modern cameras where the following conditions are met:

1. uniform temporal illumination of the 3D object surfaces,
2. Lambertian surface reflectance, and
3. object motion consistency through time to avoid motion discontinuity.

Equation 2.9 is a single equation in two unknowns which forms a single constraint line in velocity space (u, v) satisfied by any velocity on this line. As this constraint is not sufficient to compute the unknown motion, optical flow computation is ill-posed. If I_t is temporarily set to 0, equation 2.9 is equivalent to as the dot product of the gradient vector $[I_x, I_y]$ with the velocity vector $[u, v]$ being 0. In other words, the previous line of solution is passing through the origin (u=v=0) and has for normal vector the gradient vector. Therefore, optical flow is only available in the direction of the gradient vector. This phenomenon is referred to as the *aperture problem* [67] and implies that velocity can not be locally determined uniquely. Additional gradient constraints are thus required for a unique optical flow to be computed and this is performed in the literature by three major techniques:

- *Parametric* techniques, which assume that optical flow can be locally modelled by a parametric motion field and is usually the result of restricted motions such as a planar or affine motion model, see section 2.5.
- *Multiconstraint* techniques which combines several variants of the constant brightness constraint, see section 2.4.
- *Regularisation* techniques which constrain motion by enforcing local smoothness *i.e.* allow neighbouring pixels to influence the estimation of a pixel's optical flow, see section 2.7.

2.4 Multiconstraint techniques

Multiconstraint methods obtain extra constraints necessary to solve optical flow in equation 2.9 by taking advantage of the fact that image properties other than intensity, such as colour channels, contrast or entropy (a measure of the amount of information in the image) also satisfy

the brightness constancy model. However such constraints are often noisy and highly correlated and thus do not in general provide stable and accurate solutions. These approaches are explored in the works of many authors [89, 92]. For example Haralick *et al* [61] and Tretiak [132] employ constraints with noisy second-order partial derivatives. Woodham [140] uses images illuminated with different sources of light. Liu *et al* [89] take images at different spectra and use one optical flow constraint equation for each spectrum.

2.5 Parametric techniques

Motion information can be represented in different ways: 2D displacement, affine or other parametric transformations, 3D ego-motion *etc.* A common general framework has been suggested to parameterise motion models (see Anandan [4]). The unification of motion models within a general framework is possible because all the problems associated in estimating motion can be viewed, at least locally, from the perspective of image registration (as viewed by Tistarelli [129]). That is, given an image sequence, compute a representation of the motion field that best aligns pixels in one frame of the sequence with these in the next. The various approaches in motion parameters estimation differ only in terms of the assumptions the authors make about the spatial structure (or model) of the motion field, and choice of estimator.

Most typical optical flow techniques presume brightness constancy, which is often violated by time-dependent physical processes. These include changing surface orientation with respect to a directional illuminant, motion of the illuminant, and physical models of heat transport in infrared images. Additional models of image brightness variation are now used by many authors [64, 141, 63]. Negahdaripour [98] for example parameterises the image brightness by using *geometric* cues (based on the projection geometry, constraining the position of points in the scene in terms of the coordinates of their projections onto the image plane); and *radiometric* cues (which are tied to a large number of scene properties, including illumination condition, medium properties, sensor spectral response characteristics, as well as shape, position, and reflectance characteristics of the scene surfaces).

It has been recognised that applied motion models lead to accurate optical flow estimation as discussed by Black [19]. For example, a number of researchers have investigated the estimation of rigid body motion parameters without the prior estimation of optical flow (see Negahdaripour [99]). Since all the pixels within a region can contribute to this estimation, highly accurate results may be obtained. Although these models may be applicable under limited circumstances, they are often good approximations for a wide range of situations. For instance, the motion of the image of a planar surface under orthographic projection can be described as an affine transformation. But the same model forms a good approximation to motion in images of distant shallow surfaces under perspective projection, a situation not uncommon in many aerial image

sequences. Similarly, the motion of the image of the planar surface under perspective projection can be described by a eight-parameter quadratic transformation. Given the motion parameters vector $\mathbf{a}$ for a given set of pixels locally located at $\mathbf{x}$ with the motion model matrix $X(\mathbf{x})$ (also called transform matrix) the 2D displacement vector or optical flow is calculated from the velocity equation 2.10:

$$\dot{\mathbf{x}} = X(\mathbf{x})\mathbf{a} \tag{2.10}$$

Four motion model examples are given below with their motion matrix X: the translational, zoom, affine and planar model which corresponding motion vector $\mathbf{a}$ contains 2, 4, 6 and 8 parameters respectively. More details of these motion models are given in chapter 3.

- translational model: $X(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- zoom model: $X(\mathbf{x}) = \begin{bmatrix} x \\ y \end{bmatrix}$
- affine model: $X(\mathbf{x}) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}$
- planar model: $X(\mathbf{x}) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{bmatrix}$

In parametric techniques, the problem of computing optical flow is reformulated to one of parameter estimation in image *neighbourhoods* of pixels and, after linearisation of the model, standard linear regression can be used to estimate the motion parameters. A neighbourhood of pixels $\mathcal{R}_{\mathbf{x}}$ consists of a group of pixels located within a square bounding box centred on the pixel located at $\mathbf{x} = [x, y]$ which motion is to be determined. The least-squares regression approach, for example, attempts to minimise the functional $\epsilon(u, v)$, the sum of the squared errors given by the constant brightness equation in equation 2.9 by all the pixels of the neighbourhood $\mathcal{R}$:

$$\epsilon(u, v) = \sum_{\mathbf{x}' \in \mathcal{R}_{\mathbf{x}}} e(\mathbf{x}', u, v)^2 \tag{2.11}$$

where $\mathbf{x}'$ denotes the index of the neighbouring pixels. The new constant brightness equation is written as an error function as follows

$$e(\mathbf{x}, u, v) = I_x u + I_y v + I_t \tag{2.12}$$

Minimising ϵ is equivalent to solving a set of quadratic equations governed by the neighbouring pixel brightness gradients. The functional of equation 2.11 becomes minimal when the optical flow measurement at each pixel gives a minimum error term. Hence in the least-squares regression method, each pixel of a neighbourhood of pixels will contribute equally to the minimisation of E_d. In the case where all pixels of R belong to the same moving object in the image, the least-squares regression method is the most appropriate method. Pixels situated in the

neighbourhood are now considered to undergo different motions due to multiple motions in the 3D scene, or noise is introduced. These situations are very likely to happen in image sequences. In this case, the least-squares regression method will attempt to satisfy the constraints brought by all the pixels inside R simultaneously. The violations introduced by these pixels, also called *outliers* give an erroneous optical flow estimate for the central pixel. Data are becoming outliers if they do not fit any model represented by the dominant data distribution. The least-squares method hence lacks robustness in the presence of outliers whereas robust statistical techniques, introduced in the next section 2.6, can overcome such problems as implemented by Black [17] and in chapter 4.

2.6 Robust statistics

The field of robust statistics (investigated by Bunke *et al* [28] and Rousseeuw *et al* [110]) has developed methods to address the fact that the parametric models of classical statistics are often approximations of the phenomena being modelled. In particular, the field addresses how to handle outliers, or large errors, that do not conform to the statistical assumptions as explained in the works of different authors [18, 114, 136]. The main goals of robust statistical methods are to,

1. describe the structure best fitting the bulk of the data and,
2. identify deviating data points (outliers) or deviating substructures for further treatment, if desired.

Specifically, robust estimation addresses the problem of finding the values of the parameters that best fit a model to a set of data measurements in cases where the data differs statistically from the model assumptions. To state the issue more concretely, robust statistics addresses the problem of finding the values for the parameters, $\mathbf{a}$ that provides the best fit of the model $\mathbf{u}(d_i;\mathbf{a})$ to a set of n data measurements $\mathbf{d} = [d_1,\ldots,d_n]$. A robust estimator is in general defined as ρ and the goal in robust statistics is to find the values for the parameters $\mathbf{a}$ that minimise the size of the residual errors $(d_i - \mathbf{u}(d_i;\mathbf{a}))$:

$$\mathbf{a}^* = \text{argmin}_a \sum_i \rho(d_i - \mathbf{u}(d_i;\mathbf{a}), \sigma_i) \tag{2.13}$$

where σ_i is a scale parameter [17]. The minimisation process of equation 2.13 can also be referred to as a Maximum-likelihood estimation process which make this estimator a so-called M-estimator. The choice of the function ρ selects the *robustness* of a particular estimator which refers to its insensitivity to outliers or deviations from the assumed statistical model. Beauchemin *et al* [16], for example, analyse image motion in the frequency space with respect to motion discontinuities and surface translucence. The authors derive by means of models of constant and linear optical flow, the frequency structure of motion discontinuities due to

occlusion. The *median* technique is deployed in chapter 4 to discriminate optical flow outliers from the dominant motion in neighbourhood of pixels.

2.7 Regularisation techniques

Most current techniques for recovering optical flow exploit two constraints on the image motion: *data conservation* and *spatial coherence*. The data conservation constraint is derived from the observation that surfaces generally persist in time and hence the intensity structure of a small region in one image remains constant over time, although its position may change. The spatial coherence constraint embodies the assumption that surfaces have spatial extent and hence neighbouring pixels in an image are likely to belong to the same surface.

Since the motion of neighbouring points on a smooth rigid surface changes gradually, a smoothness constraint can enforce the motion of neighbouring points in the image plane. The smoothness constraint of Horn and Schunck used by the authors [68, 3, 96], for example, minimises the square of the velocity gradient over the image. Black [17] uses robust statistics to minimise the error terms given by the differences in motion estimates. Estimates obtained at points of non-zero intensity gradient are propagated iteratively over the image. Tretiak *et al* [132] regularise motion by applying Gaussian smoothing to their motion field. Weber and Malik [136] present a common regularisation method for computing optical flow using a robust version of the gradient-based least-squares framework presented in section 2.5.

2.8 Initialisation techniques

The goal in motion analysis is essentially to estimate the 2D motion fields from a given set of two consecutive frames of a sequence of images. Unless an initial estimate is provided, the estimator is expected to fail in certain conditions. For example, it is difficult to estimate large motions accurately by gradient-based optical flow techniques. There exist three main initialisation procedures, which are designed to give a quick and rough first estimate of what the motion is expected to be so that a gradient-based optical flow technique can perform accurately thereafter. These three initialisation procedures are:

- feed-forward
- block-matching
- hierarchical

The feed-forward technique bases its estimation on a short history of the previous motion estimates. The simplest case would be to project the previous motion estimate. More developed techniques might use $\alpha - \beta$ predictors (using weighted averages of previous estimates), *Kalman* filtering predictors (described in Appendix B) or projection of the motion model parameters

derived from the previous frame. Although this technique is quick, its reliability and efficiency rely directly on the accuracy of the previous estimates and the motion constancy of the 3D objects.

The second approach, the block-matching technique is a correlation-based technique which only estimate translational vector as initial estimate as described in section 2.2. These techniques can involve expensive search of grey-levels matches and are usually more expensive than the feed-forward techniques.

Arguments for use of the third hierarchical (or pyramid based) estimation techniques (see Burt [29]) have usually focused on issues of computational efficiency. A matching process that must accommodate large displacements can be very expensive to compute. Intuition suggests that if large displacements can be computed using a low-resolution version of the image, great savings in computation time may be achieved. Full-resolution images can then be used to improve the accuracy of displacement estimation by incrementally estimating small displacements. However, it can also be argued that it is not only efficient to avoid the high-resolution image when dealing with large displacements but necessary. This is because of *aliasing* of components of high spatial frequency undergoing large motion. Aliasing is the source of false matches in correspondence solutions or local minima in the objective function used for minimisation. Minimisation or matching in a multi-resolution framework helps to eliminate problems of this type. Another way of expressing this is to say that many sources of non-convexity that complicate the matching process are not stable with respect to scale. Thus the motivation for using hierarchical processing is twofold: to eliminate false matches by using 'large scale' structures, and to achieve a computationally efficient estimation.

Chapter 3

Gradient-based optical flow

Image motion generates pixel grey-level changes which can be measured by the optical flow technique. More specifically, the gradient-based estimator briefly reviewed in the previous chapter is implemented. Optical flow estimation relies on the constant brightness equation (CBE) which states that brightness at any location in space does not change over small time intervals. An optical flow estimator based on this constraint is developed in section 3.3 around the assumption of small motion magnitudes. This single constraint results in an ill-posed estimation problem referred to as the aperture problem. Additional constraints are typically provided by fitting a motion field to local neighbourhoods of pixels. Motion fields can be modelled by different models as explored in section 3.2 (which gives four examples). The planar motion model is implemented in the optical flow estimator developed here. A least-squares regression technique is then applied to minimise the error function associated with the CBE.

A non-iterative and iterative optical flow estimator are evaluated in section 3.5 and 3.6 against ground-truth data provided by a motion compensation technique described in section 3.4. The first experiments in section 3.5 show that both estimators can only estimate small motion accurately. Large motion can be relatively well estimated if a smoothing operation is performed with a Gaussian operator with kernel width the size of the motion magnitude. Therefore, large motions cannot be estimated automatically without knowledge of their magnitudes. The iterative estimator is shown to estimate a larger range of small motions (of maximum 4 pixels), and is more accurate than the non-iterative technique. A minimum of 10 iterations with at least 100 constraining points in the neighbourhood are necessary for the flow results to converge to a final and accurate estimate. The second set of experiments conducted in section 3.6 evaluates several estimators differing only in their implementation of an image gradient term. The results show that the traditional gradient approximation used in the literature is an accurate and computationally efficient version of the true expression and shall be used as the default term for the rest of this study. Dense optical flow algorithms are known to be computationally expensive. For example, it takes approximately 15 minutes to compute the motion of 576x768 pixels (using a traditional Pentium III). Computational times are reported in section 3.7. In

addition, complexity analysis demonstrates how the cost involved in computing optical flow largely depends on the number of neighbouring pixels to which the motion model is fitted.

3.1 Review

Gradient-based optical flow techniques recover motion vectors using spatio-temporal variations of the image intensity, as performed by numerous authors [4, 10, 17, 136]. Spatio-temporal variations do not always occur when objects are moving, causing difficulties in estimating optical flow (as explained for example by Horn [67]). For example, some of the variations in the scene intensity may not be due to motion, but may be caused by variations in the lighting environment as discussed in the work of Negahdaripour and Horn [99]. Nevertheless, the first assumption that is often made is that the intensity of a point in the scene is conserved, and that all variations in the image sequence are due to motion. The conservation of these assumptions gives rise to the commonly used constant brightness equation, or CBE (introduced in section 2.3) and developed in this chapter. The CBE is a single equation in two unknowns which forms a single constraint line in velocity space satisfied by any velocity on this line. As this constraint is not sufficient to compute both components of the optical flow, the computation is ill-posed. That is to say, only the motion component in the direction of the local gradient of the image intensity function may be estimated. This phenomenon, known as the aperture problem [67] implies that at any pixel the velocity can not be determined uniquely. The aperture problem is addressed by three alternate techniques: multi-constraint, parametric and regularisation techniques reviewed in sections 2.4, 2.5 and 2.7 respectively. A parametric technique is used in this chapter to increase the degree of constraint by combining the constant brightness equations with a planar motion model fitted locally to neighbourhoods of pixels.

3.2 Motion models

Objects moving in 3D scenes produce image motion when captured by a camera. The produced image motion of each individual moving object can be expressed as a 8 parameters motion model and derived using the small motion assumption. The motion parameters are expressed in terms of the 3D motion, the camera parameters and the depth of the 3D object at each pixel. In this formulation, this depth dependency means that a motion model cannot be fitted to a neighbourhood of pixels as the depth of the 3D scene is required *i.e.* the optical flow equation remains under-constrained. Section 3.2.2 eliminates this depth dependency by modelling 3D scene objects by 3D planes to allow motion fields to be locally fitted to image neighbourhoods *i.e.* the *planar motion model*. Section 3.2.3 introduces the well-known affine motion model approximation to the planar motion model, as well as for translational and zooming models derived for particular 3D motions.

3.2.1 Relationship between 3D object motion and image motion

This section describes the relationship between image motion and the relative motion between the camera and independent 3D object points. Let $\mathbf{x}$ be a point on the image plane with global coordinate $[x,\ y]^T$. The image motion of $\mathbf{x}$ is described by its velocity:

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial t} \tag{3.1}$$

A pin-hole camera model with a perspective projection model is assumed [67] and is often an acceptable model in distant imagery. Let $\mathbf{X}$ be the 3D point $[X\ Y\ Z]^T$ and $\mathbf{x}$ its projected image on the image plane. The 3D world coordinate system is centred on the optical axis at a focal length distance f behind the image plane, and the X and Y axis are parallel to the image plane axis x and y respectively as shown in figure 3.1. The Z direction coincides with the optical axis, therefore the origin of the coordinate system (x, y) is the intersection of the optical axis with the image plane. The perspective projection equation gives the following relationship between the world and the image coordinates:

$$\mathbf{x} = [x,\ y]^T \tag{3.2}$$

$$= f\left[\frac{X}{Z},\ \frac{Y}{Z}\right]^T \tag{3.3}$$

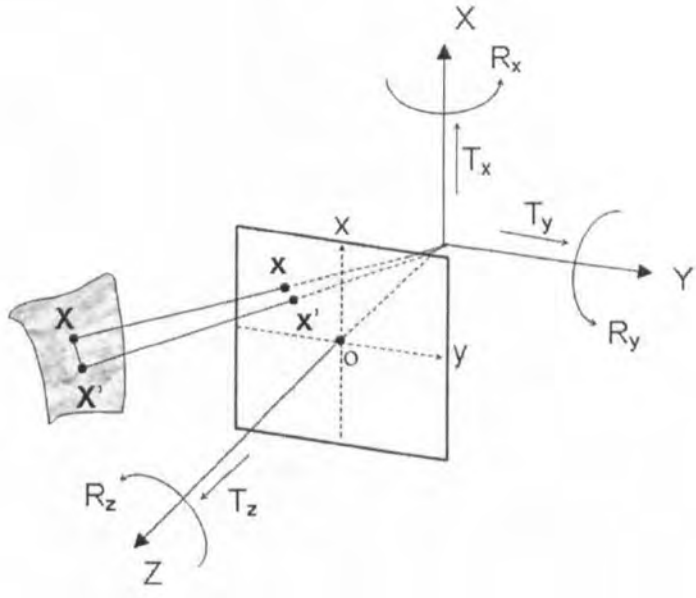

Figure 3.1: Pin-hole camera representation

Using the perspective equation 3.3, the image velocity in equation 3.1 becomes

$$\dot{\mathbf{x}} = [\dot{x},\ \dot{y}]^T \tag{3.4}$$

$$= \left[\frac{\partial x}{\partial t},\ \frac{\partial y}{\partial t}\right]^T \tag{3.5}$$

$$= \left[\frac{\partial}{\partial t}\left(f\frac{X}{Z}\right), \frac{\partial}{\partial t}\left(f\frac{Y}{Z}\right)\right]^T \tag{3.6}$$

$$= \left[\dot{f}\frac{X}{Z} + f\frac{\dot{X}}{Z} - f\frac{X}{Z^2}\dot{Z}\ ,\ \dot{f}\frac{Y}{Z} + f\frac{\dot{Y}}{Z} - f\frac{Y}{Z^2}\dot{Z}\right] \tag{3.7}$$

where $\dot{x}$, $\dot{y}$, $\dot{X}$, $\dot{Y}$ and $\dot{f}$ are the derivatives with respect to time t of x, y, X, Y and f respectively. Using the constraint that the 3D rotational angles are small, which is often true over short time intervals, then the 3D velocity can be expressed by the velocity matrix Ω followed by a translational velocity vector $\mathbf{K}$ as derived in Appendix D:

$$\dot{\mathbf{X}} = \Omega\,\mathbf{X} + \mathbf{K} \tag{3.8}$$

where $\mathbf{K} = [K_X, K_Y, K_Z]^T$ and

$$\Omega \approx \begin{bmatrix} 0 & -\Omega_Z & \Omega_Y \\ \Omega_Z & 0 & -\Omega_X \\ -\Omega_Y & \Omega_X & 0 \end{bmatrix} \tag{3.9}$$

with Ω_X, Ω_Y and Ω_Z the small angular velocities around the X, Y and Z directions respectively. Therefore, according to equation 3.8 and the small rotations assumption in equation 3.9, the 3D velocity becomes:

$$\dot{\mathbf{X}} = [\dot{X},\ \dot{Y},\ \dot{Z}]^T \tag{3.10}$$

$$\approx \begin{bmatrix} \Omega_Z Y + \Omega_Y Z + K_X \\ \Omega_Z X - \Omega_X Z + K_Y \\ -\Omega_Y X + \Omega_X Y + K_Z \end{bmatrix} \tag{3.11}$$

and the corresponding image velocity of equation 3.7 using the perspective projection in equation 3.3 becomes

$$\dot{\mathbf{x}} = [\dot{x},\ \dot{y}]^T \tag{3.12}$$

where,

$$\begin{aligned} \dot{x} &= \dot{f}\frac{X}{Z} + f\frac{\dot{X}}{Z} - f\frac{X}{Z^2}\dot{Z} \\ &\approx \dot{f}\frac{X}{Z} + \frac{f}{Z}(\Omega_Z Y + \Omega_Y Z + K_X) - f\frac{X}{Z^2}(-\Omega_Y X + \Omega_X Y + K_Z) \\ &\approx \dot{f}\frac{x}{f} + \frac{f}{Z}(\Omega_Z \frac{Z}{f} y + \Omega_Y Z + K_X) - \frac{x}{Z}(-\Omega_Y \frac{Z}{f} x + \Omega_X \frac{Z}{f} y + K_Z) \\ &\approx \dot{f}\frac{x}{f} - \Omega_Z y + \Omega_Y f + K_X \frac{f}{Z} + \Omega_Y \frac{x^2}{f} - \frac{\Omega_X}{f} xy - K_Z \frac{x}{Z} \\ &\approx f\left(\Omega_Y + \frac{K_X}{Z}\right) + \left(\frac{\dot{f}}{f} - \frac{K_Z}{Z}\right) x - \Omega_Z y + \frac{\Omega_Y}{f} x^2 - \frac{\Omega_X}{f} xy \end{aligned} \tag{3.13}$$

and

$$\begin{aligned} \dot{y} &= \dot{f}\frac{Y}{Z} + f\frac{\dot{Y}}{Z} - f\frac{Y}{Z^2}\dot{Z} \\ &\approx \dot{f}\frac{Y}{Z} + \frac{f}{Z}(\Omega_Z X - \Omega_X Z + K_Y) - f\frac{Y}{Z^2}(-\Omega_Y X + \Omega_X Y + K_Z) \end{aligned}$$

$$
\begin{aligned}
&\approx \dot{f}\frac{y}{f} + \frac{f}{Z}(\Omega_Z\frac{Zx}{f} - \Omega_X Z + K_Y) - \frac{y}{Z}(-\Omega_Y\frac{Z}{f}x + \Omega_X\frac{Z}{f}y + K_Z) \\
&\approx \dot{f}\frac{y}{f} + \Omega_Z x - \Omega_X f + K_Y\frac{f}{Z} + \frac{\Omega_Y}{f}xy - \frac{\Omega_X}{f}y^2 - \frac{K_Z}{Z}y \\
&\approx f\left(-\Omega_X + \frac{K_Y}{Z}\right) + \Omega_Z x + \left(\frac{\dot{f}}{f} - \frac{K_Z}{Z}\right)y + \frac{\Omega_Y}{f}xy - \frac{\Omega_X}{f}y^2
\end{aligned} \tag{3.14}
$$

The velocity vector components in equation 3.13 and 3.14 can be expressed in terms of matrices as the product of a model matrix X and a motion model vector $\mathbf{a}$ defined in equation 3.16 and the expression of the image velocity becomes

$$\dot{\mathbf{x}} = X(\mathbf{x})\,\mathbf{a} \tag{3.15}$$

where

$$
X(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ x & 0 \\ y & 0 \\ 0 & 1 \\ 0 & x \\ 0 & y \\ x^2 & xy \\ xy & y^2 \end{bmatrix}^T \quad \mathbf{a} \approx \begin{bmatrix} f\left(\Omega_Y + \frac{K_X}{Z}\right) \\ \frac{\dot{f}}{f} - \frac{K_Z}{Z} \\ -\Omega_Z \\ f\left(-\Omega_X + \frac{K_Y}{Z}\right) \\ \Omega_Z \\ \frac{\dot{f}}{f} - \frac{K_Z}{Z} \\ \frac{\Omega_Y}{f} \\ -\frac{\Omega_X}{f} \end{bmatrix} \tag{3.16}
$$

The vector of the motion model $\mathbf{a}$ represented by the 8 parameters in equation 3.16 describes the relationship between a 3D moving object point regarding the camera coordinate system and its corresponding projected image velocity. However, the description of this 3D-2D relationship is dependent on the depth of the object point Z projected onto the image plane before motion occurs. This means a motion field cannot be fitted to a neighbourhood of pixels without the depth of the corresponding scene (which is usually unavailable). The next section shows that this depth dependency can be avoided if objects are constrained to lie on 3D surfaces.

3.2.2 Depth-independent motion field: the planar constraint

One approach to eliminating the depth dependency in the relationship between 3D motion and image motion is to use the planar constraint [1, 4]. This constraint assumes that the depth of a neighbourhood of 3D object points can be modelled locally as a planar surface, or alternatively a single and fixed 3D point captured by a camera at different locations is assumed to be moving on a planar surface. If such a planar surface exists then it can be represented by the general surface equation 3.17:

$$k_X X + k_Y Y + k_Z Z = 1 \tag{3.17}$$

where k_X, k_Y and k_Z are the surface coefficients in the X, Y and Z respectively and represent the 3D vector normal to the surface plane. Using the perspective projection equation 3.3 in

which $x = fX/Z$ and $y = fY/Z$, we can convert from world to image plane coordinates as follows

$$\frac{1}{Z} = k_X \frac{x}{f} + k_Y \frac{y}{f} + k_Z \tag{3.18}$$

where at most two of the three parameters k_X, k_Y and k_Z can be zero. Replacing the above expression into the image velocity equations 3.13 and 3.14 gives

$$\begin{aligned}
\dot{x} &\approx f\left(\Omega_Y + \frac{K_X}{Z}\right) + \left(\frac{\dot{f}}{f} - \frac{K_Z}{Z}\right)x - \Omega_Z y + \frac{\Omega_Y}{f}x^2 - \frac{\Omega_X}{f}xy \\
&\approx f\Omega_Y + fK_X\left(k_X\frac{x}{f} + K_y\frac{y}{f} + k_Z\right) + \frac{\dot{f}}{f}x - K_Z x\left(k_X\frac{x}{f} + K_y\frac{y}{f} + k_Z\right) \\
&\quad -\Omega_Z y + \frac{\Omega_Y}{f}x^2 - \frac{\Omega_X}{f}xy \\
&\approx f(\Omega_Y + K_X k_Z) + \left(\frac{\dot{f}}{f} + K_X k_X - K_Z k_Z\right)x + (K_X k_Y - \Omega_Z)y \\
&\quad + \frac{1}{f}(\Omega_Y - K_Z k_X)x^2 + \frac{1}{f}(-\Omega_X - K_Z k_Y)xy
\end{aligned} \tag{3.19}$$

and in the y direction

$$\begin{aligned}
\dot{y} &\approx f\left(-\Omega_X + \frac{K_Y}{Z}\right) + \Omega_Z x + \left(\frac{\dot{f}}{f} - \frac{K_Z}{Z}\right)y + \frac{\Omega_Y}{f}xy - \frac{\Omega_X}{f}y^2 \\
&\approx -f\Omega_X + fK_Y\left(k_X\frac{x}{f} + K_y\frac{y}{f} + k_Z\right) + \Omega_Z x + \frac{\dot{f}}{f}y - K_Z y\left(k_X\frac{x}{f} + K_y\frac{y}{f} + k_Z\right) \\
&\quad + \frac{\Omega_Y}{f}xy - \frac{\Omega_X}{f}y^2 \\
&\approx f(-\Omega_X + K_Y k_Z) + (K_Y k_X + \Omega_Z)x + \left(\frac{\dot{f}}{f} + K_Y k_Y - K_Z k_Z\right)y \\
&\quad + \frac{1}{f}(\Omega_Y - K_Z k_X)xy + \frac{1}{f}(-\Omega_X - K_Z k_Y)y^2
\end{aligned} \tag{3.20}$$

The above image velocity $\dot{\mathbf{x}}$ can be expressed in the same linear form of equation 3.15 *i.e.* equation 3.21 but with the new motion model vector $\mathbf{a}$ in defined in equation 3.22:

$$\dot{\mathbf{x}} = X(\mathbf{x})\mathbf{a} \tag{3.21}$$

where

$$X(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ x & 0 \\ y & 0 \\ 0 & 1 \\ 0 & x \\ 0 & y \\ x^2 & xy \\ xy & y^2 \end{bmatrix}^T \quad \mathbf{a} \approx \begin{bmatrix} f(\Omega_Y + K_X k_Z) \\ \frac{\dot{f}}{f} + K_X k_X - K_Z k_Z \\ K_X k_Y - \Omega_Z \\ -f(\Omega_X - K_Y k_Z) \\ K_Y k_X + \Omega_Z \\ \frac{\dot{f}}{f} + K_Y k_Y - K_Z k_Z \\ \frac{1}{f}(\Omega_Y - K_Z k_X) \\ -\frac{1}{f}(\Omega_X + K_Z k_Y) \end{bmatrix} \tag{3.22}$$

Therefore, using the planar constraint assumption, the vector of the motion model $\mathbf{a}$ (referred to as the *planar motion model*) is expressed independently to the object depth Z. The 3D motion

can now be estimated onto the image plane as a local motion field. Using a local coordinate system $\mathbf{x}'$ with origin $\mathbf{x}_o$ in the global coordinate system $\mathbf{x}$ (with origin the centre of the image plane), see equation 3.24, it can be easily shown that the image velocity is alternatively expressed by also a 8-parameters model $\mathbf{a}'$ as follows

$$\dot{\mathbf{x}} = X(\mathbf{x}')\mathbf{a}' \tag{3.23}$$

where

$$\mathbf{x} = \mathbf{x_o} + \mathbf{x}' \tag{3.24}$$

and

$$X(\mathbf{x}') = \begin{bmatrix} 1 & 0 \\ x' & 0 \\ y' & 0 \\ 0 & 1 \\ 0 & x' \\ 0 & y' \\ x'^2 & x'y' \\ x'y' & y'^2 \end{bmatrix}^T \quad \mathbf{a}' \approx \begin{bmatrix} a_1 + a_2x_o + a_3y_o + a_7x_o^2 + a_8x_oy_o \\ a_2 + 2a_7x_o + a_8y_o \\ a_3 + a_8x_o \\ a_4 + a_5x_o + a_6y_o + a_7x_oy_o + a_8y_o^2 \\ a_5 + a_7y_o \\ a_6 + a_7x_o + 2a_8y_o \\ a_7 \\ a_8 \end{bmatrix} \tag{3.25}$$

where the motion parameters $\mathbf{a} = [a_1, ..., a_8]^T$ are defined in equation 3.16. It can be noted that under the planar constraint, the image motion in x^2, y^2 and xy terms do not change *i.e.* $a'_7 = a_7$ and $a'_8 = a_8$. The pixels in the neighbourhood locally located by $\mathbf{x}'$ around $\mathbf{x}_o$ corresponding to the 3D surface (k_X, k_Y, k_Z) therefore all contribute to the estimation of the same motion parameters. Further simplifications of the motion model can be obtained (as detailed in the next section) by using furhter assumptions about the dynamic of the 3D world relative to the camera.

3.2.3 Simplified motion models

Three further simplifications of the above planar motion model are given in this section: the affine, zoom and translational motion models. These simplified models correspond to particular or restricted motions between the 3D world and the camera.

Affine model

The planar motion model defined in equation 3.25 involves quadratic terms (xy, x^2 and y^2). An affine motion model is defined by 6 parameters, and is realised if the quadratic terms are zero *i.e.* $a_7 = a_8 = 0$ in equation 3.22 are zero if:

$$\Omega_X = -K_Z k_Y \tag{3.26}$$

$$\Omega_Y = K_Z k_X \tag{3.27}$$

In other words, an affine model exists if the depth change is restricted by the change in parameters of the 3D rotation and the surface normal vector in the X and Y directions:

$$K_Z = -\frac{\Omega_X}{k_Y} = \frac{\Omega_Y}{k_X} \tag{3.28}$$

It is not easy to imagine the highly restricted surface and motion configurations which satisfy equation 3.28. However, simpler configurations which although more marginally restrictive are easier to state. The first is given when there is no translation in Z and no 3D rotation around the X and Y axes:

$$\Omega_X = \Omega_Y = K_Z = 0 \tag{3.29}$$

Hence the motion of the 3D object surface, relative to the camera image plane, is restricted to rotations around the optical axis and translations parallel to the image plane. In practice, this means angular rotations and depth changes should be small. The second configuration is found using the same translational restriction but requiring the surface and the image plane to be parallel to each other (or alternatively the optical axis and the surface normal vector are parallel). Unlike the first configuration, the motion of the 3D surface in Z is allowed to vary:

$$\Omega_X = \Omega_Y = k_X = k_Y = 0 \tag{3.30}$$

Using orthographic projection [68], the 3D world position and the corresponding 2D image position are equivalent *i.e.* $x = X$ and $y = Y$, so is their velocities: $\dot{x} = \dot{X}$ and $\dot{y} = \dot{Y}$ an affine motion model can be also obtained from the planar constraint.

Zooming operation on static scenes

The zooming operation with a camera is often used to magnify details of a specific area of a scene. For example surveillance cameras are often used to zoom into suspect packages in underground stations or parking lots. In the case where only the zooming operation of a camera is used on a static scene, only the focal length changes and the 3D motion parameters are all zero:

$$\Omega_X = \Omega_Y = \Omega_Z = K_X = K_Y = K_Z = 0 \tag{3.31}$$

The image velocity is then described by a 1-parameter model:

$$\dot{\mathbf{x}} = \frac{\dot{f}}{f}\mathbf{x} \tag{3.32}$$

where $\dot{f}$ is the amount of focal length change per unit of time *i.e.* focal velocity.

Translational model

Translational motion fields consist of pixels whose optical flow are moving in the same direction with the same magnitude *i.e.* the same motion vector. Such uniform vectors are obtained if the 3D surface moves only in the X and Y directions and is parallel to the image plane. Assuming

the focal length does not vary in time, and the rotation angles (R_X, R_Y and R_Z) and the depth changes (T_Z) are zero *i.e.*

$$\Omega_X = \Omega_Y = \Omega_Z = K_Z = k_X = k_Y = \dot{f} = 0 \tag{3.33}$$

then a translational motion model of image velocity can be expressed as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} fK_X k_Z \\ fK_Y k_Z \end{bmatrix} \tag{3.34}$$

3.2.4 Summary

Relative motions between 3D object points and a camera can be modelled by a simple pinhole camera model. The projected image of the 3D motion into the image plane of a camera referred to as image velocity is formulated in section 3.2.1. The small 3D rotation assumption is usually met with distant objects. Section 3.2.2 shows that image velocity can be expressed independently of the object depths at each pixel if 3D objects are modelled as planes. Image velocities are then modelled by the planar motion model. Further simplifications to this model are described in section 3.2.3 *i.e.* the affine, zoom and translational models which further restrict the motion of 3D surfaces. The remaining of this chapter investigates an optical flow estimator with a planar motion model fitted onto local neighbourhoods of pixels.

3.3 Optical flow formulations

In this section, an iterative gradient-based optical flow estimator is developed. The iterative term simply means that optical flow estimation converges to an accurate solution iteratively. The constant brightness equation, CBE, is expanded in the first section into a Taylor series expansion using the assumption that the motion to be estimated is small in magnitude. Optical flow is then formulated in the second section after a least-squares minimisation procedure and the last section describes different possible interpretations of the gradient term of the optical flow expression.

3.3.1 Expansion of the CBE: constant brightness equation

Let $I_t(\mathbf{x})$ be the image brightness or grey-level of an intensity function I at time t and at location $\mathbf{x}$ in the image plane. After motion, from time t to τ, this grey-level moves to a different location $\rho(\mathbf{x}, \mathbf{a})$ where $\mathbf{a}$ describes the motion. The CBE is then defined by

$$I_t(\mathbf{x}) = I_\tau(\rho(\mathbf{x}, \mathbf{a})) \tag{3.35}$$

where the warping function $\rho(\mathbf{x}, \mathbf{a})$ defined in equation 3.36 is simply $\mathbf{x}$ displaced by $\Delta\mathbf{x}$ calculated in equation 3.37 from a motion model $\mathbf{a}$ and model matrix X (described in section 3.2):

$$\rho(\mathbf{x}, \mathbf{a}) = \mathbf{x} + \Delta\mathbf{x}_t^T(\mathbf{x}, \mathbf{a}) \tag{3.36}$$

where the displacement $\Delta\mathbf{x}_t^\tau(\mathbf{x}, \mathbf{a})$ is defined from the velocity in the image plane (see previous section) :

$$\Delta\mathbf{x}_t^\tau(\mathbf{x}, \mathbf{a}) = \Delta\tau X(\mathbf{x})\ \mathbf{a} \tag{3.37}$$

Optical flow estimators aim to estimate the unknown motion vector $\mathbf{a}$ which allow the calculation of the optical flow displacement $\Delta\mathbf{x}$. For better accuracy, $\mathbf{a}$ is embedded into an iterative process where the deviation $\Delta\mathbf{a}_{k+1}$ from one iteration denoted k to the next denoted $k+1$ is assumed small:

$$\mathbf{a}_{k+1} = \mathbf{a}_k + \Delta\mathbf{a}_{k+1} \tag{3.38}$$

where $\mathbf{a}_k$ and $\mathbf{a}_{k+1}$ are the motion estimates after k and $k+1$ iterations respectively. Because we shall embed this in a least-squares formulation, the minimisation process is guaranteed to converge. The estimates $\mathbf{a}_{k+1}$ converge toward a final solution after K iterations and the final motion vector is hence a sum of small motions:

$$\mathbf{a} = \mathbf{a}_{k\to\infty} \tag{3.39}$$

$$\approx \mathbf{a}_K \tag{3.40}$$

$$\approx \mathbf{a}_0 + \sum_{k=[0:K-1]} \Delta\mathbf{a}_{k+1} \tag{3.41}$$

It will be shown in the experimental section 3.5 that the maximum number of iterations typically required to reach convergence in motion estimation is $K = 15$. Convergence is here defined in the estimation as a certain number of iterations. The initial motion is assumed unknown and set to zero before the iteration process starts: $\mathbf{a}_0 = \mathbf{0}$. Only an initialisation technique (see section 2.8) can provide such an initial estimate as demonstrated in chapter 4. Using the iterative process of equation 3.38, the CBE of equation 3.35 can be expressed as

$$I_t(\mathbf{x}) \approx I_\tau\left(\rho(\mathbf{x}, \mathbf{a}_{k+1})\right) \tag{3.42}$$

$$\approx I_\tau\left(\rho(\mathbf{x}, \mathbf{a}_k + \Delta\mathbf{a}_{k+1})\right) \tag{3.43}$$

First approximation

Assuming the deviation $\Delta\mathbf{a}_{k+1}$ to be small, the transformation function $\rho(.)$ of equation 3.43 can be expanded into its Taylor series around $\Delta\mathbf{a}_{k+1}$, and neglecting terms higher than second-order:

$$\rho(\mathbf{x}, \mathbf{a}_k + \Delta\mathbf{a}_{k+1}) \approx \rho(\mathbf{x}, \mathbf{a}_k) + \frac{\partial\rho(\mathbf{x}, \mathbf{a}_k)}{\partial\mathbf{a}_k}\Delta\mathbf{a}_{k+1} \tag{3.44}$$

to give the new CBE

$$I_t(\mathbf{x}) \approx I_\tau\left(\rho(\mathbf{x}, \mathbf{a}_k) + \frac{\partial\rho(\mathbf{x}, \mathbf{a}_k)}{\partial\mathbf{a}_k}\Delta\mathbf{a}_{k+1}\right) \tag{3.45}$$

Second approximation

If the deviation $\Delta\mathbf{a}_{k+1}$ is assumed small then the product $\frac{\partial\rho(\mathbf{x}, \mathbf{a}_k)}{\partial\mathbf{a}_k}\Delta\mathbf{a}_{k+1}$ can also be assumed small compared to the deviation $\rho(\mathbf{x}, \mathbf{a}_k)$ and a second Taylor series approximation in equation

3.45 gives a new CBE

$$I_t(\mathbf{x}) \approx I_\tau(\rho(\mathbf{x},\mathbf{a}_k)) + \frac{\partial I_\tau(\rho(\mathbf{x},\mathbf{a}_k))}{\partial \rho(\mathbf{x},\mathbf{a}_k)} \frac{\partial \rho(\mathbf{x},\mathbf{a}_k)}{\partial \mathbf{a}_k} \Delta \mathbf{a}_{k+1} \tag{3.46}$$

$$\approx I_\tau(\rho(\mathbf{x},\mathbf{a}_k)) + \varphi_\tau(\mathbf{x},\mathbf{a}_k)\Delta \mathbf{a}_{k+1} \tag{3.47}$$

where

$$\varphi_\tau(\mathbf{x},\mathbf{a}_k) = \frac{\partial I_\tau(\rho(\mathbf{x},\mathbf{a}_k))}{\partial \rho(\mathbf{x},\mathbf{a}_k)} \frac{\partial \rho(\mathbf{x},\mathbf{a}_k)}{\partial \mathbf{a}_k} \tag{3.48}$$

3.3.2 Least-squares optical flow estimator

In real captured imagery, pixel grey-level rarely keep their intensities constant and the CBE is instead expressed as an error measurement

$$e_\tau(\mathbf{x},\mathbf{a}_{k+1}) = I_t(\mathbf{x}) - I_\tau(\rho(\mathbf{x},\mathbf{a}_k)) - \varphi_\tau(\mathbf{x},\mathbf{a}_k)\Delta \mathbf{a}_{k+1} \tag{3.49}$$

$$= \Delta I_t^\tau(\mathbf{x},\mathbf{a}_k) - \varphi_\tau(\mathbf{x},\mathbf{a}_k)\Delta \mathbf{a}_{k+1} \tag{3.50}$$

and, since $\Delta \mathbf{a}_{k+1} = \mathbf{a}_{k+1} - \mathbf{a}_k$

$$e_\tau(\mathbf{x},\mathbf{a}_{k+1}) = \Delta I_t^\tau(\mathbf{x},\mathbf{a}_k) + \varphi_\tau(\mathbf{x},\mathbf{a}_k)\mathbf{a}_k - \varphi_\tau(\mathbf{x},\mathbf{a}_k)\mathbf{a}_{k+1} \tag{3.51}$$

$$= \Delta J_t^\tau(\mathbf{x},\mathbf{a}_k) - \varphi_\tau(\mathbf{x},\mathbf{a}_k)\mathbf{a}_{k+1} \tag{3.52}$$

where

$$\Delta I_t^\tau(\mathbf{x},\mathbf{a}_k) = I_t(\mathbf{x}) - I_\tau(\rho(\mathbf{x},\mathbf{a}_k)) \tag{3.53}$$

$$\Delta J_t^\tau(\mathbf{x},\mathbf{a}_k) = \Delta I_t^\tau(\mathbf{x},\mathbf{a}_k) - \varphi_\tau(\mathbf{x},\mathbf{a}_k)\mathbf{a}_k \tag{3.54}$$

$$\tag{3.55}$$

In order to optimise the motion estimation at a pixel location $\mathbf{x}$, a minimisation process is performed over a sum of squared errors given by a set of n pixels neighbouring $\mathbf{x}$ and located at $\mathbf{x}' = [\mathbf{x}'_1 : \mathbf{x}'_n]$. This minimisation is referred to as the least-squares regression technique. It is also performed over n pixels in m neighbouring frames denoted (*i.e.* over time) by $\tau = [\tau_1 : \tau_m]$ and is expressed in equation 3.56 as

$$\epsilon(\mathbf{a}_{k+1}) = \sum_{\mathbf{x}'} \sum_{\tau} e_\tau(\mathbf{x}', \mathbf{a}_{k+1})^2 \tag{3.56}$$

$$= \mathbf{e}^T\mathbf{e} \tag{3.57}$$

where

$$\mathbf{e} = \Delta \mathbf{J}_k - \Psi_k \mathbf{a}_{k+1} \tag{3.58}$$

with the following vectors $\mathbf{e}$ and $\Delta\mathbf{J}$ and matrix Ψ defines as

$$\mathbf{e} = [e_{\tau_1}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,e_{\tau_1}(\mathbf{x}'_n,\mathbf{a}_k),\ldots,e_{\tau_m}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,e_{\tau_m}(\mathbf{x}'_n,\mathbf{a}_k)]^T \tag{3.59}$$

$$\Delta\mathbf{J}_k = [\Delta J_t^{\tau_1}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\Delta J_t^{\tau_1}(\mathbf{x}'_n,\mathbf{a}_k),\ldots,\Delta J_t^{\tau_m}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\Delta J_t^{\tau_m}(\mathbf{x}'_n,\mathbf{a}_k)]^T \tag{3.60}$$

$$\Psi_k = [\varphi_{\tau_1}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\varphi_{\tau_1}(\mathbf{x}'_n,\mathbf{a}_k),\ldots,\varphi_{\tau_m}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\varphi_{\tau_m}(\mathbf{x}'_n,\mathbf{a}_k)]^T \tag{3.61}$$

The term $\varphi_\tau(\mathbf{x}', \mathbf{a}_{k+1})$ is a row vector of p parameters described in the next section 3.3.3. The optical flow $\mathbf{a}_{k+1}$ is found for the minimum $\epsilon(\mathbf{a}_{k+1})$ when its derivative with respect to $\mathbf{a}_{k+1}$ is zero:

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{a}_{k+1}} \quad \epsilon(\mathbf{a}_{k+1}) = 0 \\
&\frac{\partial}{\partial \mathbf{a}_{k+1}} \quad \left((\Delta\mathbf{J}_k - \Psi_k \mathbf{a}_{k+1})^T (\Delta\mathbf{J}_k - \Psi_k \mathbf{a}_{k+1})\right) = 0 \\
&\frac{\partial}{\partial \mathbf{a}_{k+1}} \quad \left(\Delta\mathbf{J}_k^T \Delta\mathbf{J}_k - \Delta\mathbf{J}_k^T \Psi_k \mathbf{a}_{k+1} - \mathbf{a}_{k+1}^T \Psi_k^T \Delta\mathbf{J}_k + \mathbf{a}_{k+1}^T \Psi_k^T \Psi_k \mathbf{a}_{k+1}\right) = 0
\end{aligned} \tag{3.62}$$

Both terms $\Delta\mathbf{J}_k^T \Psi_k \mathbf{a}_{k+1}$ and $\mathbf{a}_{k+1}^T \Psi_k^T \Delta\mathbf{J}_k$ are scalars and since one is the transpose of the other, equation 3.62 simplifies to

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{a}_{k+1}} \left(\Delta\mathbf{J}_k^T \Delta\mathbf{J}_k - 2\Delta\mathbf{J}_k^T \Psi_k \mathbf{a}_{k+1} + \mathbf{a}_{k+1}^T \Psi_k^T \Psi_k \mathbf{a}_{k+1}\right) &= 0 \\
-2\frac{\partial}{\partial \mathbf{a}_{k+1}} (\Delta\mathbf{J}_k^T \Psi_k \mathbf{a}_{k+1}) + \frac{\partial}{\partial \mathbf{a}_{k+1}} (\mathbf{a}_{k+1}^T (\Psi_k^T \Psi_k) \mathbf{a}_{k+1}) &= 0
\end{aligned} \tag{3.63}$$

Moreover, by definition

$$\frac{\partial}{\partial \mathbf{a}_{k+1}} \left((\Delta\mathbf{J}_k^T \Psi_k) \mathbf{a}_{k+1}\right) = \Delta\mathbf{J}_k^T \Psi_k \tag{3.64}$$

and

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{a}_{k+1}} \left(\mathbf{a}_{k+1}^T (\Psi_k^T \Psi_k) \mathbf{a}_{k+1}\right) &= \mathbf{a}_{k+1}^T \left[(\Psi_k^T \Psi_k) + (\Psi_k^T \Psi_k)^T\right] && (3.65) \\
&= 2\mathbf{a}_{k+1}^T (\Psi_k^T \Psi_k) && (3.66)
\end{aligned}$$

Using the two previous equalities 3.64 and 3.66 in the minimisation process of equation 3.63:

$$\begin{aligned}
2\mathbf{a}_{k+1}^T (\Psi_k^T \Psi_k) &= 2\Delta\mathbf{J}_k^T \Psi_k \\
(\Psi_k^T \Psi_k) \mathbf{a}_{k+1} &= \Psi_k^T \Delta\mathbf{J}_k
\end{aligned} \tag{3.67}$$

hence

$$\begin{aligned}
\mathbf{a}_{k+1} &= (\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Delta\mathbf{J}_k && (3.68) \\
&= \left(\sum_{\mathbf{x}'} \sum_{\tau} \varphi_\tau^T(\mathbf{x}', \mathbf{a}_k) \varphi_\tau(\mathbf{x}', \mathbf{a}_k)\right)^{-1} \sum_{\mathbf{x}'} \sum_{\tau} \varphi_\tau^T(\mathbf{x}', \mathbf{a}_k) \Delta J_t^\tau(\mathbf{x}', \mathbf{a}_k) && (3.69)
\end{aligned}$$

where

$$\begin{aligned}
\varphi_\tau(\mathbf{x}, \mathbf{a}_k) &= \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{a}_k} && (3.70) \\
\Delta J_t^\tau(\mathbf{x}, \mathbf{a}_k) &= \Delta I_t^\tau(\mathbf{x}, \mathbf{a}_k) - \varphi_\tau(\mathbf{x}, \mathbf{a}_k) \mathbf{a}_k && (3.71) \\
\Delta I_t^\tau(\mathbf{x}, \mathbf{a}_k) &= I_t(\mathbf{x}) - I_\tau(\rho(\mathbf{x}, \mathbf{a}_k)) && (3.72)
\end{aligned}$$

Matrix Ψ contains the $N(= nm)$ row vectors φ (defined in equation 3.61) and matrix $\Delta\mathbf{J}$ contains N scalars ΔJ (defined in equation 3.60). The term $(\Psi_k^T \Psi_k)^{-1}$ in equation 3.68 is often referred to as the *pseudo-inverse* of matrix Ψ_k. The optical flow formulation can also be expressed in terms of $\Delta\mathbf{a}_{k+1}$:

$$\mathbf{a}_{k+1} = (\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Delta\mathbf{J}_k \tag{3.73}$$

$$= (\Psi_k^T \Psi_k)^{-1} \Psi_k^T (\Delta \mathbf{I}_k + \Psi_k^T \mathbf{a}_k) \tag{3.74}$$

$$= ((\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Delta \mathbf{I}_k) + ((\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Psi_k \mathbf{a}_k) \tag{3.75}$$

$$= (\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Delta \mathbf{I}_k + \mathbf{a}_k \tag{3.76}$$

$$= \mathbf{a}_k + \Delta \mathbf{a}_{k+1} \tag{3.77}$$

where the motion parameters between iterations are

$$\Delta \mathbf{a}_{k+1} = (\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Delta \mathbf{I}_k \tag{3.78}$$

and similarly to the vector $\Delta \mathbf{J}_k$ (equation 3.60), $\Delta \mathbf{I}_k$ is a column vector containing the $N = nm$ grey-level differences given by the N neighbouring pixels (equation 3.72).

3.3.3 Interpretation of the optical flow derivatives

This section contributes to the accurate definition of the optical flow gradient term and its comparison to the approximations used in the literature. It will be shown in the experiments in the next section that given small motion magnitudes, the traditional implementation of the gradient term is a quick and acceptable approximation of the correct expression.

The general optical flow expression developed in the previous section 3.3.2 (see equation 3.69) is implemented via the calculation of two terms: $\Delta I_t^\tau(\mathbf{x}, \mathbf{a}_k)$ (see equation 3.72) and $\varphi_\tau(\mathbf{x}, \mathbf{a}_k)$ written in equation 3.79. The $\Delta I_t^\tau(\mathbf{x}, \mathbf{a}_k)$ term is an easily implemented grey-level difference whereas the interpretation and implementation of the derivative term $\varphi_\tau(\mathbf{x}, \mathbf{a}_k)$ is difficult. By further development

$$\varphi_\tau(\mathbf{x}, \mathbf{a}_k) = \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k))}{\partial \rho(\mathbf{x}, \mathbf{a}_k)} \frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{a}_k} \tag{3.79}$$

$$\approx \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k))}{\partial \mathbf{x}} \left(\frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{a}_k} \tag{3.80}$$

Using the expression of the motion displacement ($\Delta \mathbf{x} = \Delta \tau X \mathbf{a}$ in equation 3.36 and 3.37), the third term of equation 3.80 is developed into

$$\frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{a}_k} = \frac{\partial (\mathbf{x} + \Delta \tau X(\mathbf{x}) \mathbf{a}_k)}{\partial \mathbf{a}_k} = \Delta \tau X(\mathbf{x}) \tag{3.81}$$

and the derivative term $\varphi_\tau(\mathbf{x}, \mathbf{a}_k)$ becomes

$$\varphi_\tau(\mathbf{x}, \mathbf{a}_k) = \Delta \tau \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k))}{\partial \mathbf{x}} \left(\frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{x}} \right)^{-1} X(\mathbf{x})$$

$$= \Delta \tau \, \nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) \, X(\mathbf{x}) \tag{3.82}$$

where

$$\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) = \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k))}{\partial \mathbf{x}} \left(\frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{x}} \right)^{-1} \tag{3.83}$$

The new derivative term φ_τ formulated in equation 3.82 does not present any implementation problems. The implementation of the second term $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k)$ expressed in equation 3.83 is generally approximated to a much simpler expression in the optical flow literature: approximation method 5 stated below and used by the authors Horn and Schunk [68], Haussecker *et*

al [63] and Kearney *et al* [80]. For example, there exists other possible approximations of equation 3.83 where four examples are given below in methods 1 to 5 with method 0 being the non-approximated correct expression in equation 3.82:

- method 0: $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) = \frac{\partial I_\tau(\rho(\mathbf{x},\mathbf{a}_k))}{\partial \mathbf{x}} \left(\frac{\partial \rho(\mathbf{x},\mathbf{a}_k)}{\partial \mathbf{x}}\right)^{-1}$
- method 1: $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) \approx \frac{\partial I_\tau(\mathbf{x})}{\partial \mathbf{x}} \left(\frac{\partial \rho(\mathbf{x},\mathbf{a}_k)}{\partial \mathbf{x}}\right)^{-1}$
- method 2: $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) \approx \frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}} \left(\frac{\partial \rho(\mathbf{x},\mathbf{a}_k)}{\partial \mathbf{x}}\right)^{-1}$
- method 3: $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) \approx \frac{\partial I_\tau(\rho(\mathbf{x},\mathbf{a}_k))}{\partial \mathbf{x}}$
- method 4: $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) \approx \frac{\partial I_\tau(\mathbf{x})}{\partial \mathbf{x}}$
- method 5: $\nabla_\rho I_\tau(\mathbf{x}, \mathbf{a}_k) \approx \nabla I_t(\mathbf{x})$

where $\nabla I_t(\mathbf{x}) = \frac{\partial I_t(\mathbf{x})}{\partial \mathbf{x}}$. This gradient expression φ_τ is called for every neighbouring pixels of equation 3.69 hence $\mathbf{x}$ in this section represents $\mathbf{x}'$ in equation 3.69. The inverse term present in method 0, 1 and 2 is derived for a planar motion model as

$$\frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x} + \Delta\tau X(\mathbf{x})\mathbf{a}^k\right) \tag{3.84}$$

$$= I + \Delta\tau \frac{\partial}{\partial \mathbf{x}} \left(X(\mathbf{x}\mathbf{a}^k\right) \tag{3.85}$$

$$= I + \Delta\tau \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{bmatrix} [a_1, ..., a_8]^T \tag{3.86}$$

$$= I + \Delta\tau \begin{bmatrix} a_2 + 2a_7x + a_8y & a_3 + a_8x \\ a_5 + a_7y & a_6 + a_7x + 2a_8y \end{bmatrix} \tag{3.87}$$

where I is the 2x2 identity matrix. The two translational parameters of the motion model are usually the most dominant parameters, *i.e.* $\{a1, a4\} >> a_i, \forall i \neq \{1, 4\}$. The non-translational components will be shown to be less then 1 in magnitude and the inverse term will be shown negligible in the estimation process, *i.e.*

$$\frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{x}} \approx I \tag{3.88}$$

Most optical flow frameworks implement φ by the approximation method 5 which is a good approximation if a^k is close to the correct motion parameters to be estimated. This implies that the initial estimation at $k = 0$ is to be close to the right solution too. Provided equation 3.88 is valid, the correct expression in method 0 is equivalent to method 5, *i.e.*

$$\frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_\infty))}{\partial \mathbf{x}} = \nabla I_t(\mathbf{x}) \tag{3.89}$$

where $\mathbf{a}_\infty$ is the true motion model, *i.e.* $\mathbf{a}$, when the system has converged ($\mathbf{a}_K = \mathbf{a}_\infty$ - see section 3.3.1). It can be noted that when the system starts, motion estimates are usually zero and the correct expression in method 0 is the same as method 4, provided equation 3.88 is valid:

$$\frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_0))}{\partial \mathbf{x}} = \nabla I_\tau(\mathbf{x}) \tag{3.90}$$

The expression φ can be hence expected to behave differently from one method to the other as the system converges and as evaluated in section 3.6.

3.3.4 Summary

The general expression of an iterative optical flow estimator is previously developed in section 3.3 using the constant brightness equation stating grey-levels remain similar in small interval of time and using the small motion assumption. An optical flow solution can be obtained if constrains are brought to the system and are provided here by neighbourhoods of pixels to which a motion field model (explained in section 3.2) is fitted. Optimal estimation is given from a least-squares regression technique and the optical flow estimation is evaluated in sections 3.5 and 3.6 using the methodology developed in the next section 3.4.

3.4 Developing an evaluation methodology

The optical flow estimator derived in the previous section is evaluated in sections 3.5 and 3.6. This section describes the evaluation procedure. The evaluation is performed against known motion (referred to as the ground-truth data) which is introduced in section 3.4.1. Evaluative metrics are defined in the second section 3.4.2 to compare ground-truth with the estimated flow.

3.4.1 Input and ground-truth motion

Successive frames with available known motion vectors at each pixel are not provided. Consecutive frames are instead simulated by creating a frame from a reference one using a set of synthetically ground-truth vectors $\Delta\mathbf{x}^g$. A single ground-truth vector is applied to all pixels of a frame with randomly chosen orientation and with varying magnitudes in the range 0 to 10 pixels. The reference frame, which is the current frame, is denoted I_t and the next frame I_{t+1} is created by motion compensating the reference frame with the ground-truth vector $\Delta\mathbf{x}^g$:

$$I_{t+1}(\mathbf{x}) = I_t(\mathbf{x} + \Delta\mathbf{x}^g) \tag{3.91}$$

The motion compensation operation is performed by the bilinear interpolator developed in Appendix A. However, this operation introduces a slight blurring in the construction of the next frame. The reference is hence also blurred by a slight smoothing operation performed by a Gaussian smoothing operator with kernel width 0.5. Three input sequences are used in the evaluation of the optical flow estimator. The reference frames of each sequence are displayed in figures 3.2 and 3.3: the 'boy', 'lab' and the 'yos' sequence each consisting of two frames constructed with the ground-truth. Each frame contains 384x288 black and white 256 grey-levels pixels. The three sequences are chosen for their diversity: the 'boy' sequence is captured outdoor in the countryside, the 'lab' sequence is captured indoor in an office and the 'yos' sequence is a 3D view outdoor view synthetically created. The drawback associated with synthetically generated frames is that their content do not reflect the noisy process captured with real sequences.

Figure 3.2: Reference frame of the left: 'boy' and right: 'lab' sequence

Figure 3.3: Reference frame of the 'yos' sequence

3.4.2 Error definition

Because of the lack of ground-truth optical flow vectors, evaluating as well as comparing motion estimators is always an issue. Given ground-truth data, angular and magnitude errors are calculated and reported. A suitable approach would be for example to evaluate optical flow only in problematic areas, such as across edges where occlusion occur and in areas lacking grey-level spatial variations. In this application, all pixels undergo the same motion field and a novel approach is used to evaluate optical flow: a percentage of correctly estimated flows for each frame is reported as described in the following.

The estimated optical flow displacement $\Delta\mathbf{x}(\mathbf{x})$ at a particular pixel location $\mathbf{x}$ is compared to the ground-truth displacement $\Delta\mathbf{x}^g$ (described in the previous section 3.4.1) by the motion error term $d(\mathbf{x})$ expressed as

$$d(\mathbf{x}) = \|\Delta\mathbf{x}(\mathbf{x}) - \Delta\mathbf{x}^g\| \tag{3.92}$$

The mean of the motion errors M_d over the entire set of pixels belonging to the frame denoted $\mathcal{F}$ is calculated in equation 3.93:

$$M_d = \frac{1}{\mathrm{Card}(\mathcal{F})} \sum_{\mathbf{x}\in\mathcal{F}} d(\mathbf{x}) \tag{3.93}$$

where Card($\mathcal{F}$) is the total number of pixels within the frame $\mathcal{F}$, hence it is the total number of pixels in the frame. The estimator's performance may also be measured by the percentage of pixels considered accurately estimated *i.e.*

$$P_d = \frac{1}{\text{Card}(\mathcal{F})} \sum_{\mathbf{x} \in \mathcal{F}} \begin{cases} 1 & \text{if } d(\mathbf{x}) < \mathcal{T}_d \\ 0 & \text{else} \end{cases} * 100 \tag{3.94}$$

where the threshold $\mathcal{T}_d$ has been arbitrarily chosen as 0.1 pixel.

3.5 Evaluation of the least-squares optical flow estimator

The experiments of this section consist in evaluating the performance of the iterative and non-iterative estimator previously formulated in section 3.3. The evaluation procedures are described in detail in section 3.4. The first results show that the accuracy of a non-iterative least-squares estimator depends directly on the motion-dependent smoothing operation where frames are to be smoothed with a Gaussian smoothing operator with kernel width of the size of the applied motion magnitude. More accurate results are obtained when the iterative scheme is used, without the need of a pre-smoothing operation: the estimator is accurate in estimating motion up to a few pixels in magnitude. It is also shown that both the size of the neighbourhood used to constrain the constant brightness equation and the minimum number of iterations required for minimisation to be carefully chosen for best results.

3.5.1 Estimating with non-iterative optical flow: smoothing dependency

In order to illustrate the dependency of the non-iterative optical flow estimator on the smoothing operation, the estimations are performed with an over-constrained system by selecting a square size neighbourhood of 13x13 pixels. The smoothing is performed by a Gaussian kernel of width σ and the optical flow results are evaluated against ground-truth data through the calculation of the mean motion errors M_d as described in section 3.4. The mean motion results are displayed in figure 3.4 against the factor σ for the 'lab' sequence and for various applied motion magnitudes. These results show that M_d is minimum for a unique choice of σ called the optimal factor denoted σ_o:

$$\sigma_o = \text{argmin}_{\sigma > 0}\{M_d(\sigma)\} \tag{3.95}$$

Figure 3.4 show that σ_o increases with the applied motion $\Delta\mathbf{x}^g$. In addition the error M_d rises with increasing σ_o. The values of these optimal factors are displayed in the three tables 3.1, 3.2 and 3.3 for the 'lab', 'boy' and 'yos' sequence respectively. These tables also show that the best optical flow estimates are given when σ_o are approximately set equal to the applied motion. Therefore parameter selection in the non-iterative technique cannot easily be automated as it depends on the applied motion magnitude.

If optical flow was to be expressed in a 1D case, it can be easily shown that the displacement would be approximated by $\Delta x = \Delta I / \nabla I$ using the 1st order Taylor series approximation where ΔI is the temporal change in grey-level and ∇I is the spatial derivative. This optical flow expression is only valid if the intensity I takes a linear form which is only true when motion is small relative to a camera (provided the brightness constancy is respected). However, motion can happen to be large and the linearity of the temporal grey-level distribution of a pixel no longer exists and hence motion cannot be estimated from the previous optical flow expression. The convolution of a smoothing Gaussian kernel with standard deviation σ_o the size of the motion magnitude garantees the required linearity of the pixel intensities as observed throughout the results. This Gaussian kernel smoothes enough the spatial intensity content around a pixel to simulate the temporal variation of the its grey-level motion into a linear form.

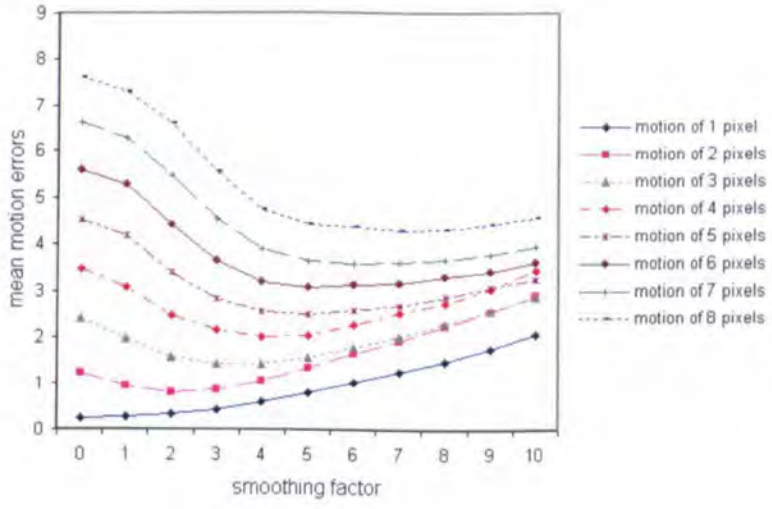

Figure 3.4: Mean motion errors M_d for various motion magnitudes with the 'lab' sequence

The percentage of pixels accurately estimated (see equation 3.94) are plotted in figure 3.5 for various applied motions for the three different sequences computed with the appropriate optimal smoothing. The results show the strong decrease of accuracy with increasing motion magnitudes: a minimum of 75% of the motions are accurately estimated for motion of 2 pixels ('lab' graph) and the percentage drops to around 50% for motion of 3 pixels in magnitude. Pixels whose motion is accurately estimated are represented as bright transparent grey-levels on the superposed frames in figure 3.6 and 3.7 for the example of the 'yos' sequence, whereas pixels which motions are non-accurately estimated are displayed as dark transparent grey-levels. These figures do not display any direct relation between the image content and the areas of correctly estimated motion vectors.

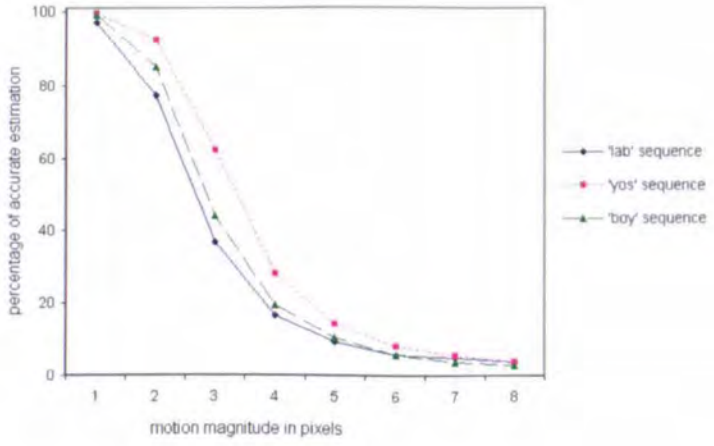

Figure 3.5: Percentage of accurately estimated motion vectors

Figure 3.6: Accurately estimated motion in the bright transparent regions and non-accurate results in the dark regions for left: 1 pixel and right: 2 pixels applied motion magnitude for the 'yos' sequence

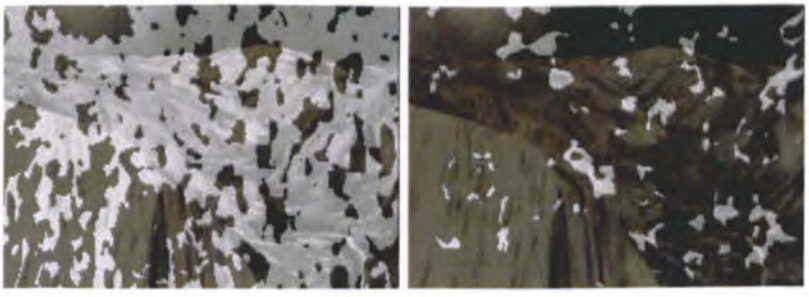

Figure 3.7: Accurately estimated motion in the bright transparent regions and non-accurate results in the dark regions for left: 3 pixel and right: 6 pixels applied motion magnitude for the 'yos' sequence

$\|\Delta \mathbf{x}^g\|$	1	2	3	4	5	6	7	8
σ_o	0	2	3	4	5	6	7	8
$M_d(\sigma_o)$	0.25	0.80	1.40	2.02	2.49	3.07	3.59	4.28

Table 3.1: Optimal smoothing factor for the 'lab' sequence

$\|\Delta \mathbf{x}^g\|$	1	2	3	4	5	6	7	8
σ_o	0	2	3	4	5	6	7	8
$M_d(\sigma_o)$	0.10	0.63	1.31	2.01	2.75	3.57	4.43	5.35

Table 3.2: Optimal smoothing factor for the 'boy' sequence

$\|\Delta \mathbf{x}^g\|$	1	2	3	4	5	6	7	8
σ_o	0	2	3	4	5	6	7	8
$M_d(\sigma_o)$	0.08	0.43	0.95	1.55	2.20	2.92	3.62	4.34

Table 3.3: Optimal smoothing factor for the 'yos' sequence

3.5.2 The iterative estimator and the neighbourhood size

The non-iterative estimator is shown in the previous section 3.5.1 to be motion-dependent. Moreover, the estimator can only estimate accurately and efficiently (in terms of percentage of successful estimations) pixels undergoing motion with maximum magnitude 1 pixel. The aim of the following experiments is to evaluate the iterative process and investigate if it can accurately estimate larger motions, and avoid the dependency of motion magnitudes.

The mean of the motion errors M_d between the estimated and the ground-truth optical flow (see section 3.4) over all the pixels of the 'yos' sequence are displayed in figures 3.8 and 3.9. The graphs in these figures are drawn with the same scale for comparisons purpose. The values of the mean errors are plotted for different iterations k and for different neighbourhood sizes. The mean error is now denoted $M_d(N,k)$ as it is evaluated against k and N. The results of these graphs show that $M_d(N,k)$ decreases rapidly until convergence as k increases. However, as the motion increases, more iterations are necessary to reach convergence and larger neighbourhood sizes are also necessary to reach the minimum converged errors.

The number of iterations required for motion estimate to convergence to a final estimate, denoted K, increases with the applied motion magnitude. For the sake of comparison, convergence is defined as the point at which the decrease in error is less than 10% of the last motion error value *i.e.*

$$K = \min_k \left\{ \frac{M_d(N,k) - M_d(N,k+1)}{M_d(N,k)} < 10\% \right\} \tag{3.96}$$

The convergence results K are displayed in table 3.4 for different neighbourhood size N with the corresponding mean motion errors $M_d(N,K)$. These data show that the system needs more

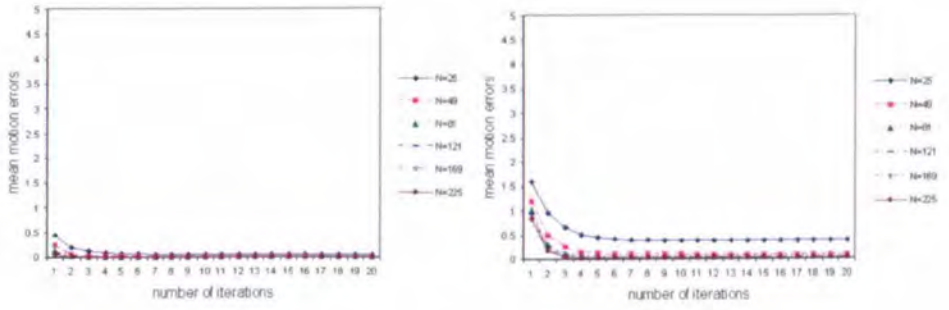

Figure 3.8: Mean motion errors of the 'yos' sequence for left: 1 pixel and right: 2 pixels motion magnitude

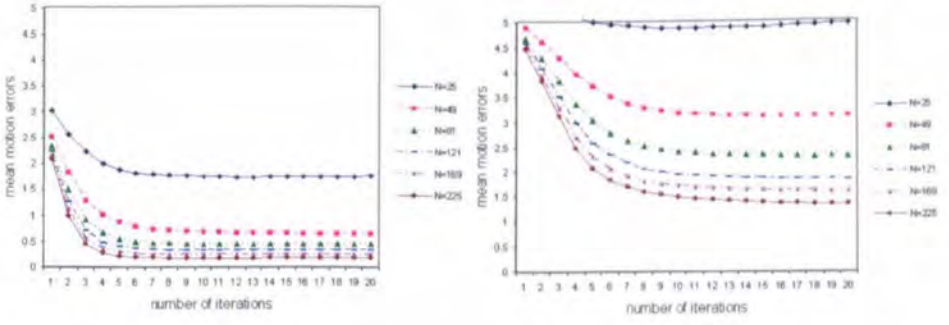

Figure 3.9: Mean motion errors of the 'yos' sequence for left: 3 pixels and right: 5 pixels motion magnitude

iterations *i.e.* K increases as the motion increases when $N \geq 81$ pixels. The corresponding error also increases with the motion and for all neighbouring sizes. The results of K in the first table show some stability when $N \geq 81$ where K take the approximate values of 4, 7, 9, 10 and 13 iterations for motion of 1, 2, 3, 4 and 5 pixels respectively. For the remaining experiments, a neighbourhood size of 100 pixels is selected.

number of iterations for convergence: K						
motion	number of constrains: N					
magnitude	25	49	81	121	169	225
1	12	15	5	4	4	4
2	11	8	6	8	6	7
3	8	11	10	9	9	9
4	7	10	10	10	10	11
5	6	10	12	12	13	14

corresponding error: $M_d(N,K)$						
motion	number of constrains: N					
magnitude	25	49	81	121	169	225
1	0.06	0.008	0.006	0.006	0.006	0.006
2	0.38	0.08	0.05	0.02	0.02	0.01
3	1.77	0.66	0.45	0.33	0.23	0.18
4	3.36	1.85	1.38	1.03	0.89	0.75
5	4.89	3.16	2.36	1.90	1.64	1.38

Table 3.4: Optimal number of iterations K and corresponding mean errors $M_d(N,K)$

3.5.3 Comparing iterative and non-iterative estimators

The percentage of optical flow vectors accurately estimated by the non-iterative and iterative estimator are displayed in figure 3.10 for the 'lab' sequence. This percentage is calculated according to equation 3.94 where a motion is accurately estimated if it does not deviate more than 0.1 pixel from the ground-truth vector. The non-iterative results are given with the optimal pre-smoothing operations and the iterative results are given after 15 iterations and with square neighbourhood of 121 pixels. The results of this figure 3.10 show clearly how the percentage of accuracy P_d increases when the iterative process is used. The results for the three sequences, the 'lab', 'yos' and 'boy' sequence are displayed in figure 3.11 using the same iterative estimator (K=15 and N=121). The accuracy behaves similarly with increasing motions. Optical flow is more efficiently estimated with first the 'yos', second 'boy' and last 'lab' sequence as observed with the non-iterative estimator. On average, the iterative estimator can estimate accurately

about 90% of the pixels motions having a magnitude not greater than 3 pixels compared with 1 with the non-iterative estimator. As additional information, a sparse set of optical flow vectors is displayed in figure 3.12 for different examples of applied motion. In this figure the white lines represent the optical flow vectors and are drawn on top of the 'boy' frame. The figures show a majority of similar vectors (same orientation and magnitude) which are the correctly estimated optical flows. These figures visually show the decrease of efficiency previously measured with the increase of the applied motion magnitude.

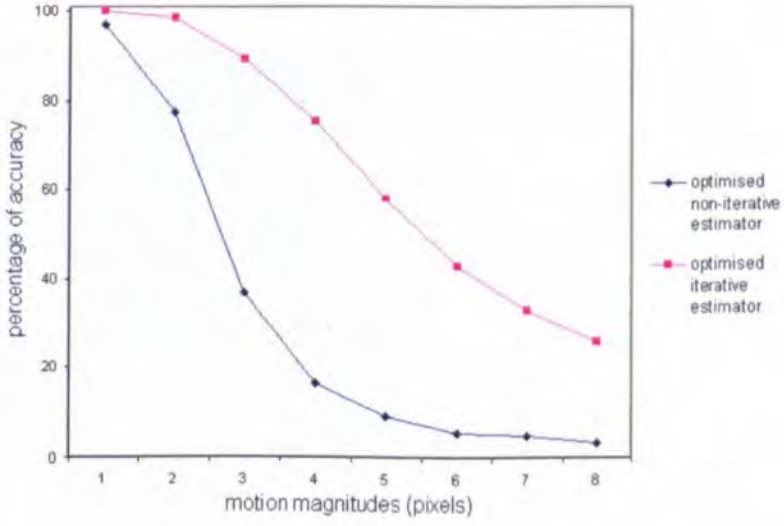

Figure 3.10: Percentage of accuracy of non-iterative and iterative estimator for the 'lab' sequence

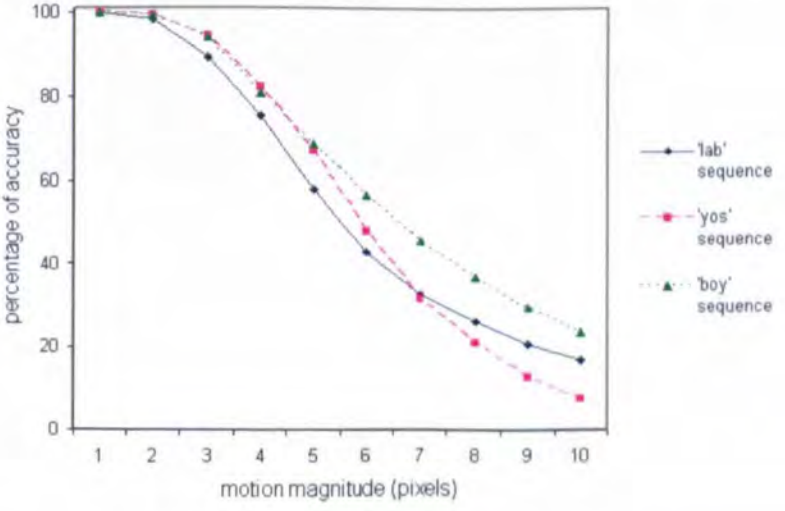

Figure 3.11: Percentage of accuracy of the iterative estimator for three sequences

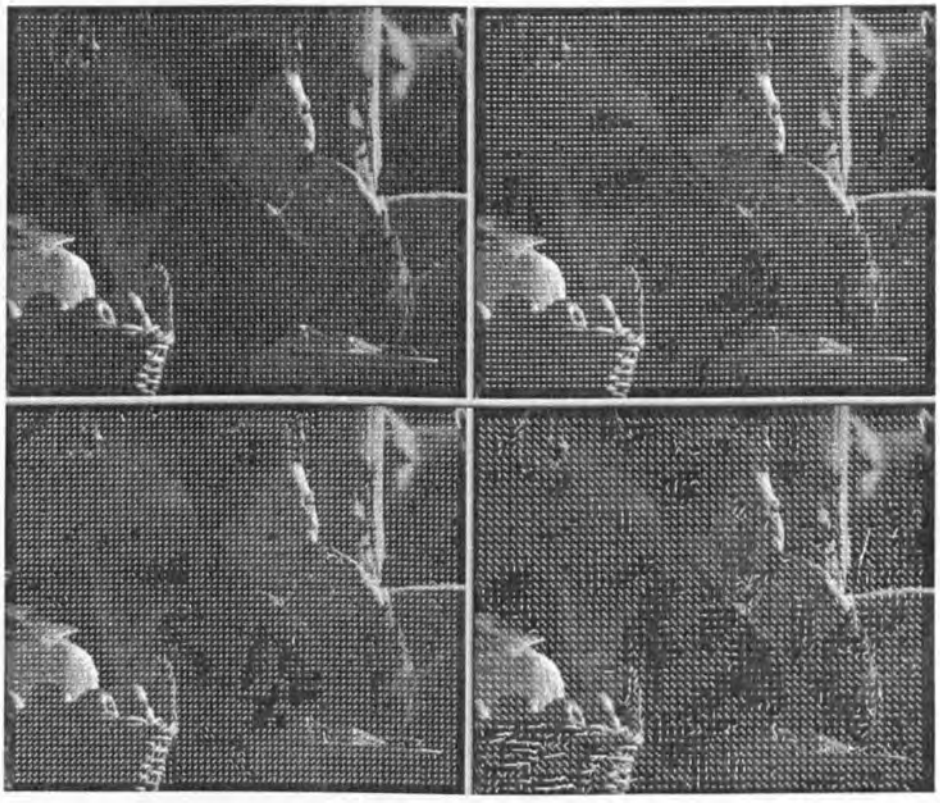

Figure 3.12: Sparse motion fields obtained by the iterative estimator (N=100 constrains and k=15 iterations) for a motion of magnitude top left: 1, top right: 2, bottom left: 2.83 and bottom right: 4.24 pixels

3.5.4 Conclusion

The experiments performed in the previous section 3.5.1 on the non-iterative estimator show that the best results are found when the smoothing factor is chosen to be equal to the unknown applied motion magnitude. Hence this estimator is heavily dependent on the motion magnitudes. Moreover, this estimator is not accurate even with optimal smoothing. Its performance decreases drastically with motion magnitude and motion with magnitude up to 1 pixel can only be accurately estimated.

The results obtained with the iterative approach presented in section 3.5.2 show that optical flow estimates converges to more accurate solutions. The optical flow accuracy also decreases with the applied motion magnitude but with a lower rate than with the non-iterative version. The parameters of the iterative technique are not dependent on prior knowledge of the motion magnitudes. It was shown that motions up to 4 pixels are accurately estimated with the iterative estimator. Its performance is also evaluated against the number of constrains and stability is reached when the constraining neighbourhood contain at least 100 pixels and when the system iterates a minimum number of 10 times.

Iterative optical flow estimation through a gradient-based approach remains an expensive dense motion estimator (as observed in a later section 3.7) and is limited to estimate accurately motion up to about 4 pixels maximum. This motion magnitude limitation is due to the constrain imposed by the small motion approximation in the optical flow development in section 3.3 but can be overcome by initialisation technique as performed in chapter 4. This chapter concludes in the next section 3.6 with the evaluation of approximated estimators differing in their implementations of the optical flow gradient or derivative term (defined in section 3.3.3).

3.6 Evaluation of the optical flow gradient terms

In the mathematical expression of the optical flow in section 3.3, the only term having a difficult interpretation and problematic implementation is the derivative term $\varphi_\tau(\mathbf{x}, \mathbf{a}_k)$:

$$\varphi_\tau(\mathbf{x}, \mathbf{a}_k) = \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k))}{\partial \rho(\mathbf{x}, \mathbf{a}_k)} \frac{\partial \rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{a}_k} \tag{3.97}$$

Implementing this gradient term involves warping the frame I_τ by a motion vector $\mathbf{a}$ of p parameters followed by the spatial derivatives of this warped image with respect to a complex warped coordinated system. Equation 3.97 is developed in section 3.3.3 where the derivative term is split into three terms. These terms can be approximated in different ways creating different algorithms with different performances.

As explained in section 3.3.3, the two translational parameters of the motion model $\mathbf{a}_k$ are usually the most dominant parameters compared to the other parameters reducing the number of potential approximations to the correct expression. As the system iterates toward the correct motion estimates, the gradient term is expected to take approximated forms.

Using the procedure described in section 3.6.1 the estimators are evaluated in section 3.6.2 with a synthetically created frame and in section 3.6.3 with a real captured frame. The results show that the quickest and simplest derivative approximation and mostly used by most authors in the literature [68, 63, 80] is an acceptable method for an optical flow estimator.

3.6.1 Evaluation criteria

The six different approximations of the derivative term φ_τ (equation 3.97) of the optical flow equation are implemented into six motion estimators whose performance is evaluated according to the following criteria:

Convergence is the K^{th} iteration at which the difference in mean motion errors between successive iterations is less than 10% than the last mean error. It should be noted that the system can reach this condition without actually converging where errors grows for later iterations.
Error is the mean motion error M_d after convergence. If convergence is met after K iterations then the error of the system is measured at the $(K+1)^{th}$ iteration *i.e.* $M_d(K+1)$.
Speed is the computational cost of the algorithm per iteration expressed in seconds with a 850 MHz Pentium III processor.

3.6.2 Results with uniformly distributed gradient synthetic data

The optical flow experiments are performed on two displaced synthetical frames which form the 'sine' sequence. A frame example is displayed in figure 3.13.

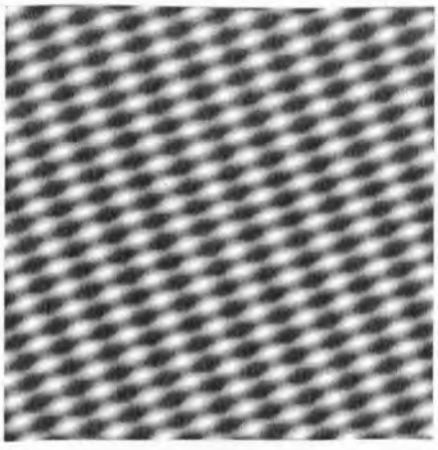

Figure 3.13: Reference frame of the 'sine' sequence

Error

The mean motion errors M_d defined in section 3.4 for the 6 estimators built are plotted in figure 3.14 and 3.15 for four different small motion magnitudes of 1, 2, 3 and 4 pixels. All the

estimators converge to small errors. Methods 0 and 3 perform similarly, as do methods 1 and 4 and methods 2 and 5. This suggests that the inverse term in equation 3.83 *i.e.* $\partial\rho(\mathbf{x}, \mathbf{a}^k)\ /\partial\mathbf{x}$ can be neglected.

The errors at convergence $M_d(K+1)$ given by the 6 different estimators are all very small for these small motion magnitudes. They slightly increase with the motion but the systems remain highly accurate for motions of up to 4 pixels applied on this synthetical frame example.

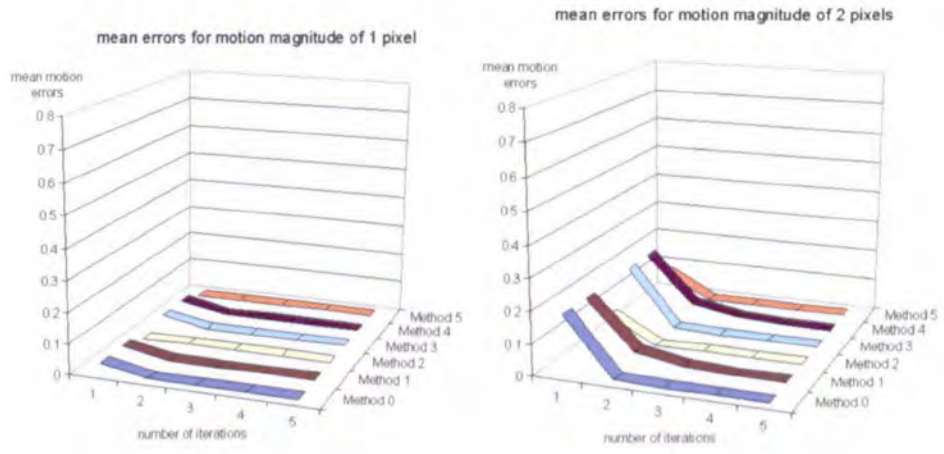

Figure 3.14: Mean motion errors for left: 1 pixel and right: 2 pixels motion magnitude

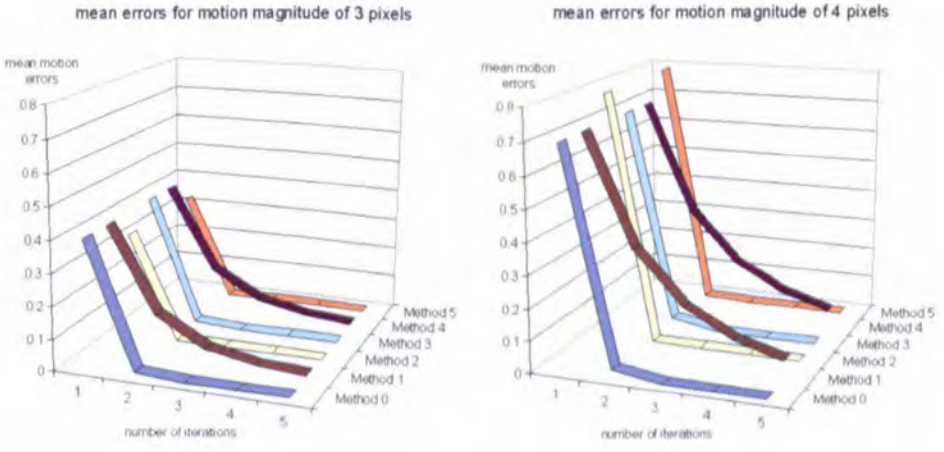

Figure 3.15: Mean motion errors for left: 3 pixels and right: 4 pixels motion magnitude

Convergence

The convergence measured by the K^{th} iteration, the mean motion error at convergence $M_d(K+1)$ and the speed of the system per iteration defined in the previous section are displayed for the six optical flow methods for maximum motion magnitudes of 2 and 4 pixels in table 3.5 and 3.6 respectively. These tables show that all six methods are accurate and convergent for motions up to 4 pixels.

All six techniques require only $K = 2$ iterations for the motion errors to converge for a motion of 2 pixels maximum, with only 1 iteration for method 2. As the motion increases, the convergence of method 0, 3 and 5 remains the same with a slight increase for method 2. However, convergence is met after more iterations for methods 1 and 4 for the larger motion.

Speed

The speed corresponding to the computational efficiency of the six methods show that methods 4 and 5 are the quickest estimators with about 4.8 seconds per frame and per iteration. Methods 1, 2 and 3 require more time between 11 and 12 seconds and method 0 is the most expensive estimator with 18.6 seconds. However, as previously described, the six techniques require different number of iterations before convergence is met. In terms of overall speed, calculated by K*Speed, method 5 is the quickest with constant speed as the motion increases. The speed of methods 0 and 3 also remains the same unlike for methods 1, 2 and 4 which speed increases with motion.

Therefore, in terms of speed, convergence and accuracy, method 5 is the best approximation of the derivative term of the optical flow formulation as also concluded through the experiments performed in the next section with a real capture frame.

optical flow	evaluation for motion of 2 pixels maximum			
method	$M_d(K+1)$	K	Speed(sec)	K*Speed
0	$7\text{x}10^{-6}$	2	18.64	37.27
1	$7.6\text{x}10^{-3}$	2	11.86	23.72
2	$5\text{x}10^{-5}$	1	11.73	11.60
3	$7\text{x}10^{-6}$	2	11.60	23.45
4	$7.6\text{x}10^{-3}$	2	4.80	9.59
5	$3\text{x}10^{-6}$	2	4.79	9.57

Table 3.5: Systems evaluation for applied motions up to 2 pixels

3.6.3 Results with real image data

The experiments run on the synthetical frame in the previous section 3.6.2 are run once more in this section but on a real captured frame from the 'lab' sequence: see figure 3.2. Six estimators using different approximations of the optical flow derivative term (listed in the five methods of section 3.3.3) are evaluated in terms of mean motion errors M_d defined in section 3.4. The previous experiments showed that methods 0, 1 and 2 have the same performances of methods 3, 4 and 5 respectively. Hence, only these last three methods are evaluated in the remaining experiments of this section.

optical flow method	evaluation for motion of 4 pixels maximum			
	$M_d(K+1)$	K	Speed(sec)	K*Speed
0	$2x10^{-4}$	2	18.637	37.274
1	$5.8x10^{-2}$	5	11.858	59.29
2	$3x10^{-6}$	2	11.727	23.192
3	$2x10^{-4}$	2	11.596	23.454
4	$5.8x10^{-2}$	5	4.797	23.985
5	$3x10^{-6}$	2	4.787	9.574

Table 3.6: Systems evaluation for applied motions up to 4 pixels

The mean motion errors of approximation method 3, 4 and 5 are displayed in the graphs of figures 3.16 and 3.17. The results given by method 4 clearly show no convergence of the errors when iterating, even for a motion magnitude of 1 pixel. It is also less accurate than the two other methods. The results given by method 3 are the most accurate and convergence is met for all motions. Convergence is also met for method 5 if the motion does not exceed 2 pixels and whose accuracy is as good as method 3's.

The previous observations made from figure 3.16 and 3.17 are resumed in table 3.7 using the error criteria M_d and the convergence criteria K developed in section 3.4. The speed criterion is not used here as it is not altered by the input sequence. In this table, **X** is inserted whenever an estimator is not converging to a final motion estimate.

The results of table 3.7 show that motion errors given by method 3 always converge with motion. However, the converged errors remains accurate for motion less than 3 pixels in magnitude *i.e.* $M_d < 0.02$ pixel but the system becomes inaccuarte for larger motion with for example about 1 pixel error for a motion of 4 pixels in magnitude. The accuracy of method 5 remains as accurate as method 3's for motion less than 3 pixels and this method does not converge for larger motions. As observed in the previous section, method 4 give the worst results and it only converges for motion less than 2 pixels in magnitude.

The percentage of correctly estimated pixel motions (see definition of P_d in section 3.4) are displayed for the three techniques in figure 3.18, confirming the lack of accuracy of method 4 and the similarity of the two other methods. As previously observed, method 4 is never an appropriate method to estimate optical flow. Method 3, the closest approximation to the correct optical flow gradient expression, performs the best with real images. However, because method 3 is more expensive (see speed results in previous section) than method 5, this latter is preferred as optical flow technique although it lacks of accuracy and becomes unstable in term of convergence for pixels having a magnitude greater than 3 pixels.

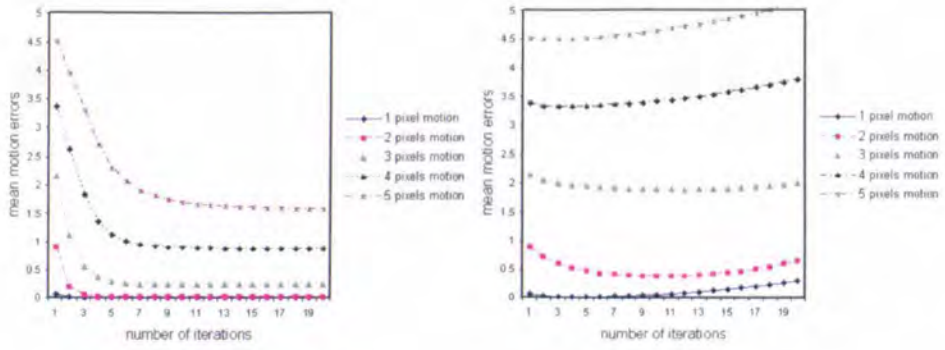

Figure 3.16: mean motion errors of left: method 3 and right: method 4 for various motions

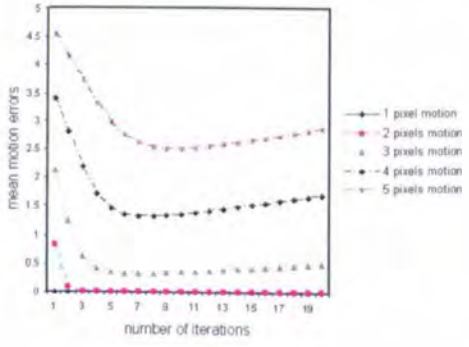

Figure 3.17: mean motion errors of method 5 for various motions

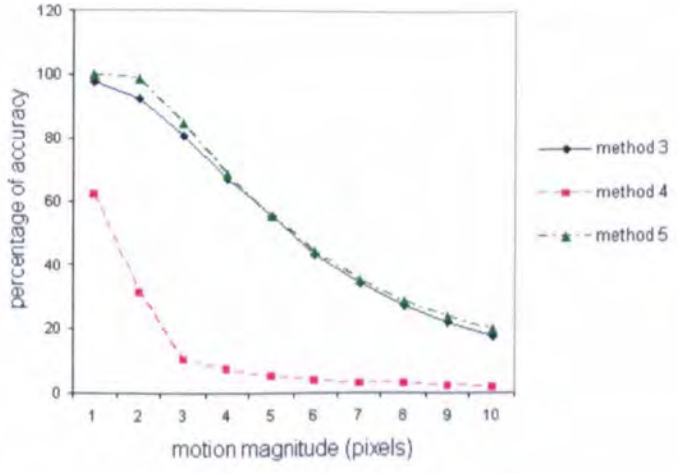

Figure 3.18: Percentage of correct optical flow estimation

method	motion magnitude (pixels)							
	1		2		3		4	
	K	M_d	K	M_d	K	M_d	K	M_d
3	3	0.06	5	0.02	7	0.2	7	0.9
4	2	0.02	X	X	X	X	X	X
5	1	0.006	4	0.02	X	X	X	X

Table 3.7: Errors M_d at convergence K given by method 3, 4 and 5 for small motions

3.6.4 Conclusion

The calculation of the optical flow expression developed in section 3.3 requires a complicated derivative term to be calculated. This term, once developed can be approximated by six different methods defined in section 3.3.3. These 6 methods mainly consist of a combination of a derivative term plus an inverse term. Their performances are evaluated in section 3.6.2 with a pair of synthetically created frames and in section 3.6.3 using a real captured frame. The results show that the inverse term can be neglected as it does not influence the estimation of the local motion field: optical flow is estimated using a small local window as neighbourhood of pixels where optical flow deviations at the borders of this window are expected to be negligible compared to the motion at the center. This side effect may be considered non-negligible in different applications such as in global motion estimation (see chapter 5). The derivative term of the approximation techniques can be successufully approximated by the commonly used term in the literature in the case where motions remain small *i.e.* they do no exceed 3 pixels in magnitude. This small motion limitation is addressed in the next chapter.

3.7 Timing estimation

Optical flow is computed at every pixel of a frame from a sum of K vectors $\Delta\mathbf{a}_{k+1}$ (with $k = [0 : K-1]$) containing p motion parameters developed (see section 3.3.2):

$$\Delta\mathbf{a}_{k+1} = \left(\sum_{\mathbf{x}'}\sum_{\tau}\varphi_\tau^T(\mathbf{x}',\mathbf{a}_k)\varphi_\tau(\mathbf{x}',\mathbf{a}_k)\right)^{-1}\sum_{\mathbf{x}'}\sum_{\tau}\varphi_\tau^T(\mathbf{x}',\mathbf{a}_k)\Delta I_t^\tau(\mathbf{x}',\mathbf{a}_k) \tag{3.98}$$

where

$$\varphi_\tau(\mathbf{x}',\mathbf{a}_k) \approx \Delta\tau\ \nabla I_t(\mathbf{x})\ X(\mathbf{x}) \tag{3.99}$$

$$\Delta I_t^\tau(\mathbf{x}',\mathbf{a}_k) = I_t(\mathbf{x}) - I_\tau(\mathbf{x}+\Delta\tau X(\mathbf{x})\mathbf{a}_k) \tag{3.100}$$

The terms $\nabla I_t(\mathbf{x})$, $X(\mathbf{x})$ and $\mathbf{a}_k$ are matrices of sizes 1×2, $2\times p$ and $p\times 1$ respectively. The term φ in equation 3.99 is given from section 3.6 and ΔI_t^τ in equation 3.100 is the motion compensated grey-level difference between time t and τ. According to equation 3.98, the time required to compute the flow $\Delta\mathbf{a}_{k+1}$, denoted $T_{\Delta\mathbf{a}}$, is split into two costs as defined in equation

3.101: the time T_{build} to compute the four sums that build the optical flow matrices and the time T_{inverse} required to perform the matrix inversion of this summation denoted T_{inverse}.

$$T_{\Delta \mathbf{a}} = T_{\text{build}} + T_{\text{inverse}} \tag{3.101}$$

The time required to build the four sums of matrices T_{build} is decomposed into two sets of two sums as shown in equation 3.98. The first set involves the construction of a square matrix of size $p \times p$ and requires $\mathcal{O}(p^2)$ operations. The second set involves the multiplication of a matrix with a scalar computed using a bilinear interpolation technique (see Appendix A) and can be shown to take $\mathcal{O}(p)$ operations. Hence the overall speed for building the matrices for $N = n \times m$ neighbouring pixels (where $\mathbf{x}' = [\mathbf{x}'_1 : \mathbf{x}'_n]$ and $\tau = [\tau_1 : \tau_m]$) requires $\mathcal{O}(Np^2)$ and can be modelled by equation 3.102 where a_1, a_2 and a_3 are constants:

$$T_{\text{build}} = \mathcal{O}(Np^2) \tag{3.102}$$

$$\approx N(a_1 p^2 + a_2 p + a_3) \tag{3.103}$$

Performing an inversion of the square matrix in equation 3.98 of size $p \times p$ requires generally $\mathcal{O}(p^3)$ operations as estimated in [103]. The SVD (Singular Value Decomposition) is used in this study to invert matrices and is modelled by equation 3.104

$$T_{\text{inverse}} = \mathcal{O}(p^3) \tag{3.104}$$

$$\approx b_1 p^3 + b_2 p^2 + b_3 p + b_4 \tag{3.105}$$

Time measurements are performed using a 850 MHz PC Pentium III with different motion models with $p = [2, 4, 6, 8]$ parameters and various number of constraining points N. The following parameters are estimated as

$$[a_1,\ a_2,\ a_3] \approx [5.87\text{x}10^{-8},\ 2.91\text{x}10^{-7},\ 1.58\text{x}10^{-6}] \tag{3.106}$$

$$[b_1,\ b_2,\ b_3,\ b_4] \approx [-3.48\text{x}10^{-8},\ 2.13\text{x}10^{-6},\ -5.69\text{x}10^{-6},\ 1.50\text{x}10^{-5}] \tag{3.107}$$

The terms a_3 and b_4 in equation 3.106 and 3.107 are zero in theory as they correspond to the time computations of T_{build} and T_{inverse} respectively with zero motion parameters and empty matrices. They are measured as non-zero due to extra matrix computations involved before estimation starts. The computational costs of T_{build} for different constrains sizes N are compared to the cost T_{inverse} in figure 3.19 for 4 different motion models ($p = [2 : 8]$). The graphs show that T_{inverse} is more expensive if $N = 1$ than T_{build} but becomes much less expensive than T_{build} when $N > 10$ pixels. The experimental results obtained in this chapter (see section 3.5) show that a minimum number of 100 pixels in neighbourhood are necessary to constrain the optical flow estimation. Therefore, for such numbers, the optical flow estimation is dominated by the operations involved in building the matrices which takes $\mathcal{O}(N)$ for a given motion model. For example, the total optical flow computation cost of all the pixels of a 576x768 pixels frame takes approximately 15 minutes with a Pentium III using a planar motion model (of p=8 parameters - see section 3.2) and after iterating 15 times.

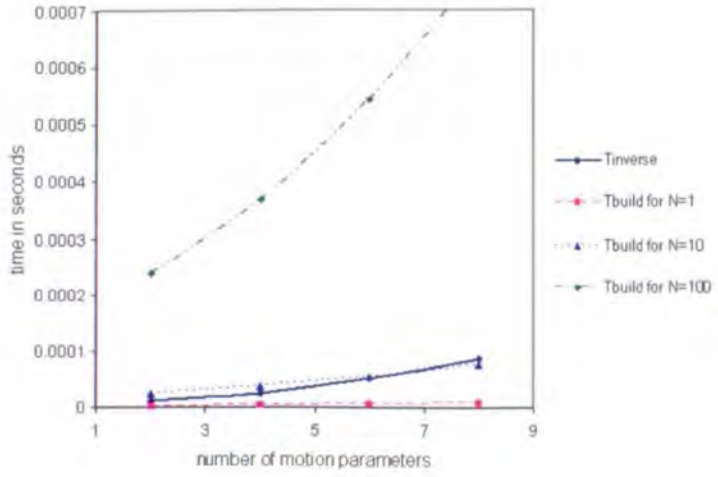

Figure 3.19: Computation cost of the inverse matrix operation T_{inverse} and the costs for building matrices T_{build} for several N

3.8 Discussion

Pixel velocity is estimated in this chapter by the gradient-based optical flow technique developed in section 3.3. It is based on the small motion assumption using the CBE (constant brightness equation) which states that grey-levels remain similar in small interval of time. The optical flow equation is constrained by fitting local motion field around each pixel, represented by square neighbourhoods of pixels. The dynamic of the 3D world in a camera coordinate system shows that the motion field can be modelled by a planar model described in section 3.2. The optical flow estimator is evaluated in section 3.5 which demonstrates that an iterative estimation approach will converge to a minimum solution. However, the estimator accuracy is limited to small applied motion magnitudes. Moreover, the estimation usually requires more than 100 pixels in the neighbourhoods for best results and a minimum of 10 iterations per pixel, which makes the estimation computationally expensive.

A study of the gradient term of the optical flow equation shows in section 3.6 that it can be approximated in the case of local motion estimation by a simple gradient term widely used in the optical flow literature. Although this approximation improves greatly the speed of the algorithm, the optical flow estimator remains a very expensive technique in providing dense optical flow fields as observed in section 3.7.

The motion estimator implemented in this chapter can estimate small motion accurately when motion models can be fitted into the motion field of neighbourhood of pixels. However, in real imagery, pixels are likely to move with high velocity and local distributions of pixels are likely to contain pixels undergoing different motion fields. In this case, the neighbourhood may contain multiple motion fields. Moreover, grey-levels noise often occurs in real images caused for example by the optical device, objects transparencies or changes of the scene illumination

which results in the violations of the small motion and the constant brightness assumptions. The next chapter 4 investigates two solutions to improve the optical flow estimator: a robust statistics technique to deal with noise and multiple motions and a hierarchical technique to deal with high pixel motions.

Chapter 4

Addressing limitations of optical flow

An optical flow technique was described in chapter 3 to estimate motion displacements at each pixel between frames. The estimation of small motion is successful on low noise images using the least-squares regression estimator. Importantly, all pixels in each neighbourhood required to constrain the optical flow equation must undergo the same motion: the motion of the central pixel of this neighbourhood. Under such experimental conditions, the results show that motions with maximum magnitude 2 to 3 pixels are accurately estimated. However, low noise images with uniform motions are unlikely to occur in real captured datasets. This chapter concentrates first on developing a robust method to deal with neighbourhoods containing alternate amounts of noise. The second focus of this chapter is to improve the estimator in estimating larger motion magnitudes by the use of a hierarchical technique.

Real-life image sequences contain various sources of noises. Change of illumination, non-uniform reflective properties of 3D surfaces, transparencies, high-frequency noise from the optical system are some of the examples of sources of errors disrespecting the constant brightness equation (defined in section 3.3). Moreover, real-image data is likely to contain objects moving with different motion models and hence introduce noise (referred to as *motion noise*) within pixels neighbourhoods. These sources of noise are dealt with a technique known as *robust statistics* explained in section 4.1. The experimental results show that such techniques can retrieve the correct information with the presence of high levels of noise, whereas the least-squares approach is sensitive to any level of noise. Despite the efficient discrimination of grey-level noise, however, robust statistics cannot discard the total influence of outlying motions at object boundaries where occlusion occurs.

Gradient-based optical flow estimators have a limited range of operation in terms of motion amplitude. Chapter 3 shows the optical flow estimator is limited to estimate accurately only small motions. However, if an initialisation technique is provided, a first estimate of the motion

would make the system iterate toward the true motion. Such initialisation techniques could comprise of the feed-forward, block-matching and hierarchical techniques as reviewed in section 2.8. The hierarchical method is explored in this chapter. It is based on a coarse-to-fine approach referred to as the *Laplacian pyramid* technique which estimates accurately larger motions at the coarsest level allowing a better estimation of motions in finer levels, as explained in section 4.2. The results show that the maximum estimable motion relies on the size of the frame at the coarsest level. However, resolution decreases as frames are sub-sampled creating aliasing in the motion estimation process. Hence, the smallest motion estimation is limited to a resolution issue. The results show that for image sizes of about 300 to 400 pixels in lateral dimension, minimum 10 pixels motions can be accurately estimated.

Dense fields of motion vectors are of great interest in many applications ranging from cartography, medical applications, structure recovery, depth assigning *etc.* However, optical flow estimators are shown in this chapter to be inaccurate at edge pixels where several motions occur, also at pixels lacking local grey-level spatial variations and at pixels undergoing too large motions. Therefore, a confidence annotated to each pixel would render the flow field useful for other applications. Section 4.3 describes a way to provide an uncertainty measurement of the optical flow estimates based on covariance expressions.

4.1 Robust statistics

The least-squares optical flow estimator implemented in chapter 3 squares all the residual terms of each pixel within the spatio-temporal window *i.e.* the neighbouring pixels: they all contribute equally in the minimisation process of the motion flow. When no noise is present in images and when the grey-levels of the neighbouring pixels correspond to the light reflected by the same object undergoing a unique 3D motion relative to the camera, all estimator would perform the same and optimally. However, these assumptions are never always true as grey-level noise is likely to occur as well as *motion noise* due to occluding objects in the scene. The least-squares scheme is for example an optimal estimator when images contain zero-mean white noise in their grey-level distributions.

In order to better cope with such problems, a technique based on robust statistics is implemented. An overview of robust statistics is given in section 4.1.1 and described in section 4.1.2. The *median* robust statistical method is chosen as the estimator in this study for its computational speed and its accuracy and is evaluated versus the least-squares technique in section 4.1.4. The details of the experimental procedures are given in section 4.1.3. This chapter concludes with a demonstration that robust statistical methods are able to cope with a high percentage of grey-level and/or motion noise while the least-squares accuracy greatly decreases with increasing levels of noise.

4.1.1 Review

The least-squares estimator is the most common regression estimator used in the development of optical flow solutions. However, residuals given by several motion fields within the same neighbourhood do not belong to a single normal distribution modelled by the residuals given by the most dominant motion. Such outlying errors are referred to as motion outliers. Noise caused by the optical system or the changes in the lighting environment also produces outliers and are referred in general as grey-level outliers. An alternative estimator is therefore required to reduce the influence of outliers. Such estimators are known as robust statistical estimators [28, 110, 18, 114, 136]. Under suitable conditions, successive iterations of a robust estimator can successfully eliminate all contributions from outliers and therefore model the motion of the intended region only. Robust estimators can successfully deal with significantly large proportions of outliers as reviewed in section 2.6. The robust median estimator is used in this chapter for its efficiency and speed. It belongs to the family of M-estimators because it is a maximum likelihood-based estimator.

4.1.2 Robust statistical estimators

The residuals given by the constant brightness equation of each neighbouring pixel in the optical flow estimation (see section 3.3.2) are all equally squared in the minimisation process. A robust regressor instead weights the residuals according to their deviations from a normal distribution, as expressed in equation 4.1 where sample index i represents the location of the neighbourhood pixel in the spatio-temporal window used in the previous chapter.

$$\epsilon(\mathbf{a}) = \sum_{i=1,N} \gamma(e_i(\mathbf{x}_i, \mathbf{a})) \tag{4.1}$$

where γ is the robust estimator kernel function, N is the number of samples and $\mathbf{a}$ is the unknown motion vector. The latter is estimated when the sum of errors ϵ is minimum, or when its derivative with respect to $\mathbf{a}$ is zero:

$$\frac{\partial \epsilon(\mathbf{a})}{\partial \mathbf{a}} = 0 \tag{4.2}$$

$$\sum_{i=1,N} \frac{\partial \gamma(e_i(\mathbf{x}_i, \mathbf{a}))}{\partial \mathbf{a}} = 0 \tag{4.3}$$

$$\sum_{i=1,N} \frac{\partial \gamma(e_i(\mathbf{x}_i, \mathbf{a}))}{\partial e_i(\mathbf{x}_i, \mathbf{a})} \frac{\partial e_i(\mathbf{x}_i, \mathbf{a})}{\partial \mathbf{a}} = 0 \tag{4.4}$$

$$\sum_{i=1,N} \Gamma(e_i(\mathbf{x}_i, \mathbf{a})) \frac{\partial e_i(\mathbf{x}_i, \mathbf{a})}{\partial \mathbf{a}} = 0 \tag{4.5}$$

where $\Gamma(e)$ is referred to as the *influence function* of error e and defined as

$$\Gamma(e) = \frac{\partial \gamma(e)}{\partial e} \tag{4.6}$$

The second term of equation 4.5, the derivative of the error term, is developed in section 4.1.2 and the first term, the influence function, is set by the chosen robust estimator. This influence

function has to somehow weight each residual term so that the sum of the weighted errors belonging to outliers is minimal. The influence function indicates how much influence a residual e_i has on the estimator, and dividing it by the residual itself gives the weight associated with the current error e_i as expressed in equation 4.8. Therefore the weighting term, rather than the influence function, is to be calculated and the estimation problem of equation 4.5 is equivalent to a weighted least-squares estimation:

$$\sum_{i=1,N} \left(w(e_i(\mathbf{x}_i, \mathbf{a}))e_i(\mathbf{x}_i, \mathbf{a}) \frac{\partial e_i(\mathbf{x}_i, \mathbf{a})}{\partial \mathbf{a}} \right) = 0 \tag{4.7}$$

where weights w are defined from errors e and their influence Γ by

$$w(e) = \Gamma(e)/e \tag{4.8}$$

A robust statistical estimator is designed by choosing the desired characteristics of the influence function, $\Gamma(e_i(\mathbf{x}_i, \mathbf{a}))$. The median estimator is chosen among a variety of robust estimators for its simplicity of implementation, efficiency and computational speed [28, 131]. The characteristics of the least-squares and robust median estimators are compared in the next two sections.

The least-squares estimator

Using a least-squares estimator, residuals influence the estimation process proportionally to their intensities. Zero residuals do not influence the estimator efficiency, however as the residual term increases in value, the efficiency of the estimator decreases. Figure 4.1 shows three graphs of the least-squares regression technique: the kernel function on the left graph introduced in equation 4.1, the influence function in the middle graph defined in equation 4.6 and the weighting function on the right graph defined in equation 4.8. This figure demonstrates that all errors are all equally weighted hence making the least-squares technique highly sensitive to outliers [17].

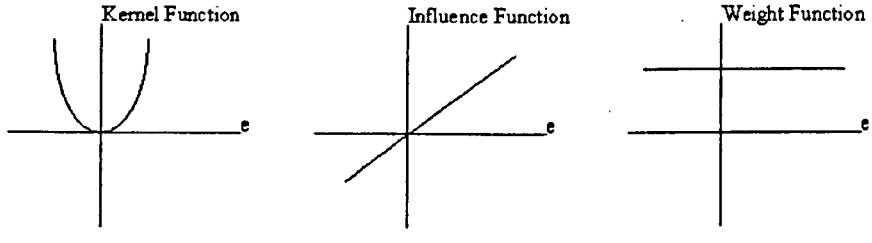

Figure 4.1: Least-squares estimator characteristics

The median estimator

The median regression technique is based on a simple idea: all errors equally influence the regression technique. This results in small errors being largely weighted and vice versa. Figure 4.2 displays the kernel (introduced in equation 4.1), influence (defined in equation 4.6) and

weight function (defined in equation 4.8) of the median estimator. Comparing graphs 4.1 and 4.2, a single outlier can offset the optical flow estimation using a least-squares regressor whereas the median estimator can deal with a dataset containing outliers.

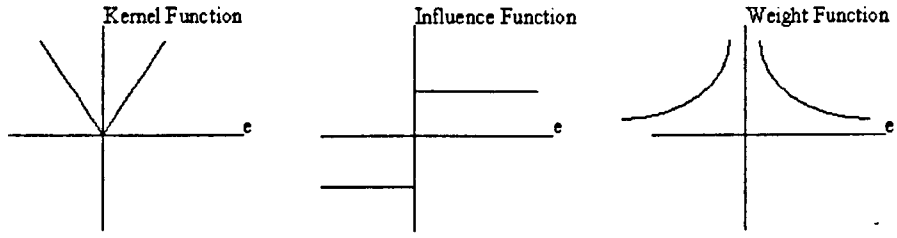

Figure 4.2: Median estimator characteristics

In general, the experimental errors e_i may be in any units of measurement; so in order to decide if a residual is considered *large* it is compared to an estimate of the error scale. This scale has to be robust itself so it depends only on the *good* data (inliers) and does not get blown up by the outliers. Thus, not only the regression coefficients has to be estimated in a robust way but also a scaling factor denoted σ^*. For Gaussianly distribution errors, the scale estimate is calculated as follows [131]:

$$\sigma^* \approx 1.4826 \times \text{Median}_i\left(|e_i|\right) \tag{4.9}$$

This preliminary scale estimate in equation 4.9 is then used to determine the weight w associated with each residual of the distribution:

$$w_i = \begin{cases} \frac{1}{e_i} & \text{if } \left|\frac{e_i}{\sigma^*} \leq 2.8\right| \\ 0 & \text{else} \end{cases} \tag{4.10}$$

There does not exist any robust estimators capable of rejecting more than 50% of outliers within a distribution. For example, the median estimator can eliminate successfully outliers if they do not represent more than approximately 30% [131], as experimented in section 4.1.4.

Implementation of a robust estimator

The new functional for a robust statistical estimator to be minimised is given by equation 4.1 and adopted in this section for an iterative estimation where the motion vector between the k^{th} and $k+1^{th}$ iteration is denoted $\mathbf{a}_{k+1}$. Neighbourhood of pixels contain n pixels located at $\mathbf{x}' = [\mathbf{x}'_1 : \mathbf{x}'_n]$ and between m surrounding frames indexed by $\tau = [\tau_1 : \tau_m]$, as described in the previous chapter 3 (section 3.3.1):

$$\epsilon(\mathbf{a}_{k+1}) = \sum_{\mathbf{x}'} \sum_{\tau} \gamma\left(e_\tau(\mathbf{x}', \mathbf{a}_{k+1})\right) \tag{4.11}$$

with motion compensated grey-level error $e_\tau(\mathbf{x}', \mathbf{a}_{k+1})$ corresponding to the constant brightness equation derived in chapter 3 (equation 3.52 in section 3.3.1):

$$e_\tau(\mathbf{x}', \mathbf{a}_{k+1}) = \Delta J_i^\tau(\mathbf{x}, \mathbf{a}_k) - \varphi_\tau(\mathbf{x}, \mathbf{a}_k)\mathbf{a}_{k+1} \tag{4.12}$$

where

$$\Delta J_t^\tau(\mathbf{x}, \mathbf{a}_k) = \Delta I_t^\tau(\mathbf{x}, \mathbf{a}_k) - \varphi_\tau(\mathbf{x}, \mathbf{a}_k)\mathbf{a}_k \quad (4.13)$$

$$\varphi_\tau(\mathbf{x}, \mathbf{a}_k) = \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k)}{\partial \mathbf{a}_k} \quad (4.14)$$

Finding the motion vector $\mathbf{a}_{k+1}$ so the functional ϵ is minimum is equivalent to setting its derivative to zero as performed in the early section 4.1.2. Repeating this procedure with the robust functional in equation 4.11 leads to this new minimisation process

$$\sum_{\mathbf{x}'}\sum_{\tau}\left(w(e_\tau(\mathbf{x}', \mathbf{a}_{k+1}))e_\tau(\mathbf{x}', \mathbf{a}_{k+1})\frac{\partial e_\tau(\mathbf{x}', \mathbf{a}_{k+1})}{\partial \mathbf{a}_{k+1}}\right) = 0 \quad (4.15)$$

Or, with the simplified notations of equations 4.20 to 4.24:

$$\sum_{\mathbf{x}'}\sum_{\tau}\left(w\, e\frac{\partial e}{\partial \mathbf{a}}\right) = 0 \quad (4.16)$$

$$\sum_{\mathbf{x}'}\sum_{\tau}\left(w(\Delta J - \varphi\,\mathbf{a})\frac{\partial}{\partial \mathbf{a}}(\Delta J - \varphi\,\mathbf{a})\right) = 0 \quad (4.17)$$

$$\sum_{\mathbf{x}'}\sum_{\tau} w(\Delta J - \varphi\,\mathbf{a})(-\varphi) = 0 \quad (4.18)$$

$$\sum_{\mathbf{x}'}\sum_{\tau} w\Delta J\varphi - \sum_{\mathbf{x}'}\sum_{\tau} w\,\varphi\,\mathbf{a}\varphi = 0 \quad (4.19)$$

where

$$e \leftrightarrow e_\tau(\mathbf{x}', \mathbf{a}_{k+1}) \quad (4.20)$$

$$w \leftrightarrow w(e_\tau(\mathbf{x}', \mathbf{a}_{k+1})) \quad (4.21)$$

$$\Delta J \leftrightarrow \Delta J_t^\tau(\mathbf{x}, \mathbf{a}_k) \quad (4.22)$$

$$\varphi \leftrightarrow \varphi_\tau(\mathbf{x}, \mathbf{a}_k) \quad (4.23)$$

$$\mathbf{a} \leftrightarrow \mathbf{a}_{k+1} \quad (4.24)$$

Using the fact that $\varphi\mathbf{a}\varphi = (\varphi^T\varphi\mathbf{a})^T$, the minisation is further developed into

$$\sum_{\mathbf{x}'}\sum_{\tau} w\Delta J\varphi = \sum_{\mathbf{x}'}\sum_{\tau} w\varphi\mathbf{a}\varphi \quad (4.25)$$

$$\left(\sum_{\mathbf{x}'}\sum_{\tau} w\varphi\Delta J\right)^T = \left(\sum_{\mathbf{x}'}\sum_{\tau} w\varphi\mathbf{a}\varphi\right)^T \quad (4.26)$$

$$\sum_{\mathbf{x}'}\sum_{\tau} w\varphi^T\Delta J = \sum_{\mathbf{x}'}\sum_{\tau} w(\varphi\mathbf{a}\varphi)^T \quad (4.27)$$

$$\sum_{\mathbf{x}'}\sum_{\tau} w\varphi^T\Delta J = \sum_{\mathbf{x}'}\sum_{\tau} w\varphi^T\varphi\mathbf{a} \quad (4.28)$$

The motion vector $\mathbf{a}$ can be then estimated by

$$\mathbf{a} = \left(\sum_{\mathbf{x}'}\sum_{\tau} w\varphi^T\varphi\right)^{-1}\sum_{\mathbf{x}'}\sum_{\tau} w\varphi^T\Delta J \quad (4.29)$$

and using the original notation from equations 4.20 to 4.24, the motion estimate $\mathbf{a}_{k+1}$ is estimated by

$$\mathbf{a}_{k+1} = \left(\sum_{\mathbf{x}'}\sum_{\tau} w(e_\tau(\mathbf{x}', \mathbf{a}_{k+1}))\,\varphi_\tau(\mathbf{x}, \mathbf{a}_k)^T\varphi_\tau(\mathbf{x}, \mathbf{a}_k)\right)^{-1} \quad (4.30)$$

$$\sum_{\mathbf{x}'}\sum_{\tau} w\left(e_\tau(\mathbf{x}', \mathbf{a}_{k+1})\right) \varphi_\tau(\mathbf{x}, \mathbf{a}_k)^T \Delta J_t^\tau(\mathbf{x}, \mathbf{a}_k) \tag{4.31}$$

$$= \left(\sum_{\mathbf{x}'}\sum_{\tau} \varphi'_\tau(\mathbf{x}, \mathbf{a}_k)^T \varphi_\tau(\mathbf{x}, \mathbf{a}_k)\right)^{-1} \sum_{\mathbf{x}'}\sum_{\tau} \varphi'_\tau(\mathbf{x}, \mathbf{a}_k)^T \Delta J_t^\tau(\mathbf{x}, \mathbf{a}_k) \tag{4.32}$$

where

$$\varphi'_\tau(\mathbf{x}, \mathbf{a}_k) = w\left(e_\tau(\mathbf{x}', \mathbf{a}_{k+1})\right) \varphi_\tau(\mathbf{x}, \mathbf{a}_k) \tag{4.33}$$

The expression of the optical flow vector $\mathbf{a}_{k+1}$ in equation 4.32 after $k+1$ iterations can also be expressed in terms of matrices as follows

$$\mathbf{a}_{k+1} = \left(\Psi_k'^T \Psi_k\right)^{-1} \Psi_k'^T \Delta \mathbf{J}_k \tag{4.34}$$

with

$$\Delta \mathbf{J}_k = \left[\Delta J_t^{\tau_1}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \Delta J_t^{\tau_1}(\mathbf{x}'_n, \mathbf{a}_k), \ldots, \Delta J_t^{\tau_m}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \Delta J_t^{\tau_m}(\mathbf{x}'_n, \mathbf{a}_k)\right]^T \tag{4.35}$$

$$\Psi_k = \left[\varphi_{\tau_1}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \varphi_{\tau_1}(\mathbf{x}'_n, \mathbf{a}_k), \ldots, \varphi_{\tau_1}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \varphi_{\tau_1}(\mathbf{x}'_n, \mathbf{a}_k)\right]^T \tag{4.36}$$

$$\Psi'_k = \left[\varphi'_{\tau_1}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \varphi'_{\tau_1}(\mathbf{x}'_n, \mathbf{a}_k), \ldots, \varphi'_{\tau_m}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \varphi'_{\tau_m}(\mathbf{x}'_n, \mathbf{a}_k)\right]^T \tag{4.37}$$

The optical flow expression of equation 4.34 is very similar to the least-squares estimator in chapter 3 (equation 3.68). The only difference being that the robust estimator weights its matrix Ψ^T. Robust optical flow can also be expressed between iterations, as achieved in equation equation 3.78:

$$\Delta \mathbf{a}_{k+1} = (\Psi_k'^T \Psi_k)^{-1} \Psi_k'^T \Delta \mathbf{I}_k \tag{4.38}$$

where $\Delta \mathbf{I}_k$ is a vector of motion compensated errors $\Delta I_t^\tau(\mathbf{x}', \mathbf{a}_k)$ (see section 3.3.1) with 2D displacement $\Delta \mathbf{x}$ defined from the chosen motion model $\mathbf{a}$ and its associated matrix transform $X(\mathbf{x})$ (descibed in section 3.2):

$$\Delta \mathbf{I}_k = \left[\Delta I_t^{\tau_1}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \Delta I_t^{\tau_1}(\mathbf{x}'_n, \mathbf{a}_k), \ldots, \Delta I_t^{\tau_m}(\mathbf{x}'_1, \mathbf{a}_k), \ldots, \Delta I_t^{\tau_m}(\mathbf{x}'_n, \mathbf{a}_k)\right]^T \tag{4.39}$$

$$\Delta I_t^\tau(\mathbf{x}, \mathbf{a}_k) = I_t(\mathbf{x}) - I_\tau(\mathbf{x} + \Delta \mathbf{x}_t^\tau(\mathbf{x}, \mathbf{a}_k)) \tag{4.40}$$

$$\Delta \mathbf{x}_t^\tau(\mathbf{a}_k) = \Delta t X(\mathbf{x}) \mathbf{a}_k \tag{4.41}$$

Problem of the robust statistical optical flow

Estimating optical flow $\mathbf{a}^{k+1}$ at the $(k+1)^{th}$ iteration given the known motion $\mathbf{a}^k$ at the previous iteration k presents a chicken-egg problem: it requires computing weights from the unknown motion $\mathbf{a}^{k+1}$. Hence, optical flow is not determinable unless it is assumed that differences in error e between iterations are small and

$$w(e_\tau(\mathbf{x}', \mathbf{a}_{k+1})) \approx w(e_\tau(\mathbf{x}', \mathbf{a}_k)) \tag{4.42}$$

can be replaced in equation 4.33 to become

$$\varphi'_\tau(\mathbf{x}, \mathbf{a}_k) \approx w\left(e_\tau(\mathbf{x}', \mathbf{a}_k)\right) \varphi_\tau(\mathbf{x}, \mathbf{a}_k) \tag{4.43}$$

Conclusion

Robust statistics introduces a kernel function to the minimisation process of the optical flow estimation in order to reject outliers introduced by noise or by non-unique motion field occurring in neighbourhood of local pixels. The weighting kernel of the least-squares scheme is compared to the one of the median scheme. The least-squares is proven non-tolerant to any outliers whereas the median can prevent a distribution of errors to be contaminated by large percentages of outliers. The median estimator is chosen as robust regressor and its implementation is equivalent to a weighted least-squares estimator, where pixels are weighted according to their motion-compensated grey-level errors.

4.1.3 Evaluation procedure

Robust statistics can cope with two sources of noise: grey-level noise introduced for example by the optical device, scene transparencies or changes in the lighting environment and motion noise introduced when pixels undergo motions different than the dominant one. Detecting the first source of noise is rather difficult but is easily eliminated by a robust estimator as this noise usually does not contaminate a high percentage of pixels unlike motion noise which can contaminate a large percentage of pixels at edge pixels where occlusion occurs. Occlusion is simulated in the first section which also describes how motion estimates are evaluated against ground-truth data. The evaluating results are presented in section 4.1.4 for the least-squares and median estimators.

Input and ground-truth motion

Motion is estimated between two frames in a synthetical and real captured sequence called the 'sine-sine' and 'texture-yos' sequence respectively: see figure 4.3. The 'sine-sine' sequence is created so all the frames contain enough texture or grey-level content for best performance of the estimator. The estimator will be then evaluated with textures reflecting the textures of the real world with the 'texture-yos' sequence. A sequence is created from a reference frame using ground-truth motion fields, therefore there does not exist any high frequency noise but only motion noise is introduced.

A horizontal and uniform motion field is applied on the right half of a reference frame whereas a zero motion field is applied on the left half. The right hand side pixels are therefore undergoing varying translational motion fields. We aim to investigate the accuracy of the optical flow estimator across motion boundary which is why this latter remains fixed between pair of frames. The moving right hand side frames are created by manually cutting canvas from a bigger and same image. Therefore, as grey-level moves, new pixels arise from either the left or right hand side of the right-hand frames wether the motion goes in the right or left direction respectively. All the frames contains 384x288 black and white 256 grey-levels pixels.

Given a frame I_t, the frame I_{t+1} is created using the bilinear interpolation technique described in Annex A.2 to motion compensate the reference frame I_t according to

$$I_{t+1}(\mathbf{x}) = I_t(\mathbf{x} + \Delta\mathbf{x}^g(\mathbf{x})) \tag{4.44}$$

where $\Delta\mathbf{x}^g$ represents the 2 ground-truth motion vectors separated by a fixed motion boundary situated at half the width of the frame Width/2:

$$\Delta\mathbf{x}^g(\mathbf{x}) = \begin{cases} [0,\ \Delta y^g]^T & \text{if } y > \text{Width}/2 \\ [0,\ 0]^T & \text{else} \end{cases} \tag{4.45}$$

where $|\Delta y^g|$ is either 0 or > 0 pixels.

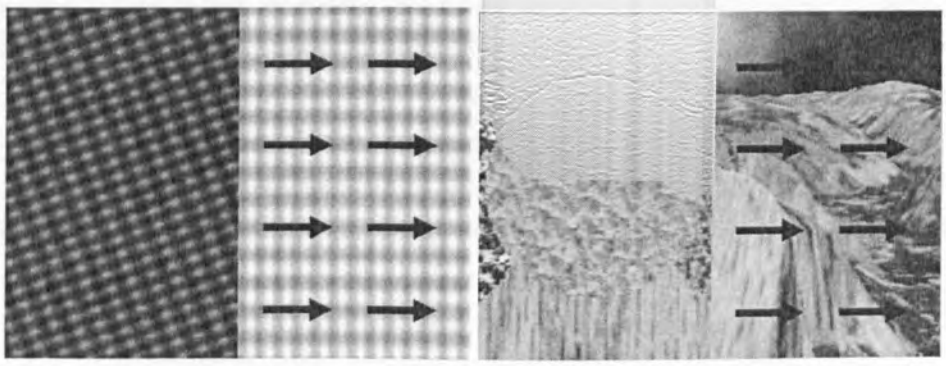

Figure 4.3: Frame example of the left 'sine-sine' and right: 'texture-yos' sequence

Vertical mean motion errors

Each estimated optical flow vector $\Delta\mathbf{x}(\mathbf{x})$ at location $\mathbf{x}$ is compared to the ground-truth $\Delta\mathbf{x}^g(\mathbf{x})$ defined in equation 4.45 through the motion error term $d(\mathbf{x})$:

$$d(\mathbf{x}) = \|\Delta\mathbf{x}(\mathbf{x}) - \Delta\mathbf{x}^g(\mathbf{x})\| \tag{4.46}$$

The estimator's performance is evaluated across the fixed motion boundary by measuring the mean M_d of the motion errors $d(\mathbf{x})$ given by all the pixels located in a parallel direction to the boundary's direction. Because the motion boundary is vertical, the errors are averaged along x for a fixed y as

$$M_d(y) = \frac{1}{H}\sum_{x_i=1}^{H} d(x_i, y) \tag{4.47}$$

where H is the height of the frame, or the number of pixels along it. The M_d values are plotted for the ten columns on either side of the true motion boundary to investigate the accuracy at pixel neighbourhoods which include this boundary.

4.1.4 Evaluation of a robust statistical estimator

Chapter 3 provides an iterative least-squares optical flow technique which accurately estimates pixels undergoing small motion magnitudes of 3 pixels maximum. For larger increasing motions,

the efficiency of the estimator decreases rapidly in terms of percentage of accurately estimated flows. This least-squares estimator is compared in this section with the robust median estimator previously developed in section 4.1.2 for small motions. They are implemented using the following optimal parameters: a square neighbourhood of 289 (17*17) pixels has been chosen between 2 frames (a minimum number of 100 pixels is necessary) and the system is iterating 15 times.

The mean vertical motion errors $M_d(y)$ defined in section 4.1.3 are plotted for a motion magnitude example of 1 pixel in figure 4.4 and for a motion of 3 pixels in figure 4.5 and using the 'sine-sine' input sequence. The horizontal axis is referred to as the edge axis where positive locations correspond to the right motion and the negative or zero values correspond to the left motion (set to zero in these examples). The motion boundary is therefore located between edge pixel location 0 and 1. The results of these two graphs display two different error distributions for the two different motion magnitudes where the motion errors increase progressively near the motion boundary. This is particularly evident for the large motion magnitude for both estimators. The maximum left motion error does not vary significantly along the edge axis: a maximum of 0.35 pixel for the least-squares and 0.22 pixel for the robust technique. Unlike the zero motion field, the right motion errors are maximum when the contamination is the highest (location 1 on the edge axis) and of the order of the applied motion magnitude for both estimators. Nevertheless, the robust motion estimates are visibly more accurate across the edge axis and for both motion magnitudes.

The accuracy of the least-squares and robust median estimators are measured by a thresholding operation on the mean motion errors: a motion at a pixel location y is considered accurately estimated if its corresponding motion error $M_d(y)$ is less than a threshold arbitrarily set to 10% of the maximum M_d given by the pixels belonging to the same edge than this pixel. The range of pixel locations across the edge, where the least-squares and robust estimators are measured inaccurate are displayed in table 4.1 for 4 different right motions of the 'sine-sine' sequence.

	estimators	
motion (pixels)	least-squares	robust median
1	[-7:8]	[-3:3]
2	[-7:8]	[-3:3]
3	[-7:8]	[-2:4]
4	[-7:8]	[-2:4]

Table 4.1: Range of edge pixel locations where the least-squares and robust estimators are estimated inaccurate

The results of table 4.1 show constancy in the range of edge pixels where the estimators are inaccurate. However, the robust estimator performs better than the least-squares around the edge, as shown by the smaller ranges. The least-squares estimation is not contaminated by a

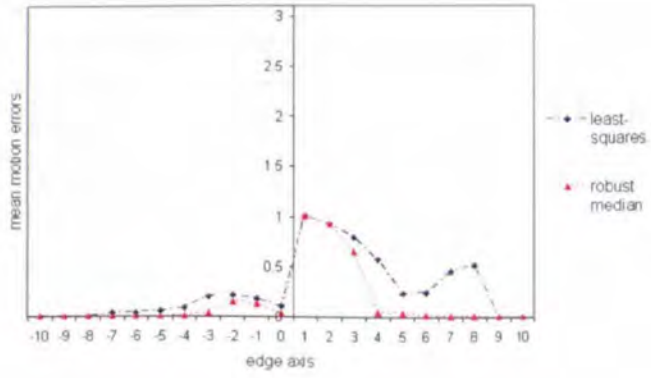

Figure 4.4: Mean motion errors of the least-squares and robust estimators for 1 pixel motion with the 'sine-sine' sequence

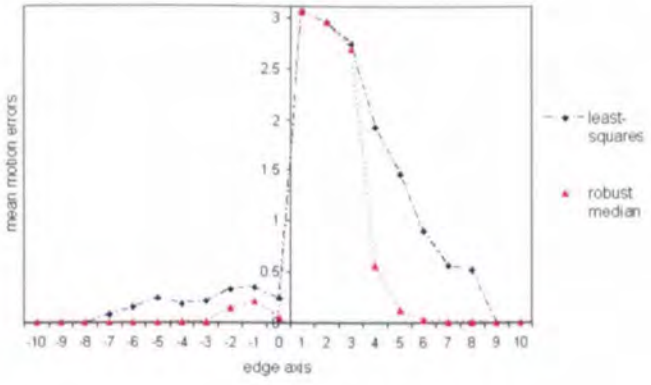

Figure 4.5: Mean motion errors of the least-squares and robust estimators for 3 pixels motion with the 'sine-sine' sequence

motion noise for pixels outside the range $[-7:8]$ which correspond to a level of 6% for both left and right contamination. In the robust case, the estimator is not contaminated for edge pixels outside the range $[-3:3]$ for motion less or equal than 2 pixels which correspond to an approximate level of contamination of 26.5% on the left side and 32.5% on the right side. For larger motions, between 3 and 4 pixels, the robust estimation is not contaminated for the range of location $[-2:4]$ which correspond to the opposite levels of contamination found for smaller motions. Hence, the robust range slightly shifts toward the right on the edge axis for larger motion greater than 2 pixels: the estimation is less contaminated along the direction of the applied motion. Similar results are observed when using the 'texture-yos' sequence in figure 4.6 which display the estimated mean vertical displacements in the x and y directions. The left applied motion is $\Delta\mathbf{x}^g = [0,0]^T$ and the right motion is $\Delta\mathbf{x}^g = [0,-2]^T$. These figures show a distinct improvement of accuracy of the robust estimator compared to the least-squares technique.

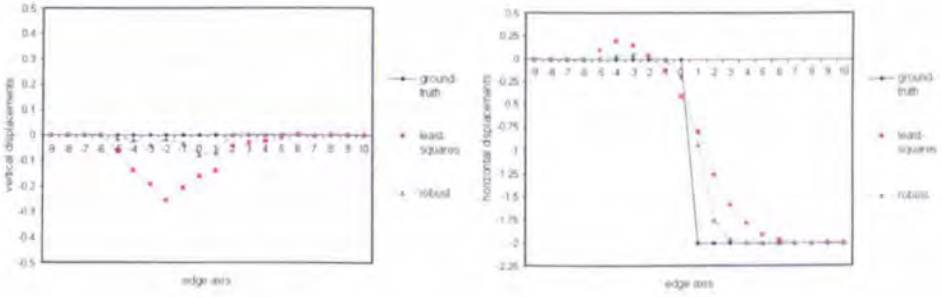

Figure 4.6: Least-squares and robust motion estimates on left: the vertical and right: horizontal direction against ground-truth across the edge with the 'texture-yos' sequence

4.1.5 Conclusion

Robust statistics introduced in section 4.1.2 shows how an adaptive kernel can weight the majority of outliers introduced for example by high frequency noise or motion noise occurring at edge pixels where occlusion is likely to occur. These outliers are weighted in a way that they do not participate in the motion estimation process. The median technique is chosen as robust estimator for its computational speed and great efficiency [131]. This estimator is compared in section 4.1.4 to the least-squares estimator implemented in section 3. The results show that the median estimator improves significantly the accuracy around edge pixels containing various levels of outliers while being as accurate as the least-squares for neighbourhood of pixels containing no outliers.

Robust estimators are accurate if the percentage of contamination inside neighbourhood of pixels does not overpass a threshold delimited by its kernel. The maximum level of contamination which the estimator can deal with is slightly influenced by the applied motion. In general,

the robust estimator is shown to be capable of dealing with approximately 30% contaminated motion. For higher levels given around edges, motion cannot be accurately estimated by a single robust statistical technique unless more information about the motion is provided.

4.2 Hierarchy: addressing large motions using Laplacian pyramids

As observed in chapter 3, gradient-based motion estimators generally fail to estimate motions. This limitation can be addressed by appropriate initialisation techniques which provide and initial estimate of the flow. The three main initialisation procedures used in the literature and introduced in section 2.8 are designed to give a quick and rough first estimate of what the motion is expected to be so the motion estimator can perform accurately thereafter. These three techniques are: the feed-forward, block-matcher and hierarchical techniques as reviewed in the first section 4.2.1.

In this study, motion is analysed between pair of frames independently of the results obtained with previously captured frames, and a spatial hierarchical technique is used to provide initial motion estimates. This technique is often implemented as a Laplacian pyramid, as it will be described in section 4.2.2. Input sequences are presented in section 4.2.4 with evaluation metrics to evaluate motion estimates against ground-truth. The evaluative results of the Laplacian estimator are given in section 4.2.5 which show how initialisation enables large motions to be accurately computed.

4.2.1 Review

Motion initialisation techniques are classified into three main categories: a temporal-based technique referred to as feed-forwarding and two spatial-based techniques referred to as block-matching and hierarchical techniques as reviewed in section 2.8. Feed-forward estimators use results located in previous frames to initialise the iterative estimator. Block-matching techniques (also called correlation-based techniques) give a translational initial motion as a first estimate by searching templates of grey-levels within relatively large search areas as described in section 2.2. This search and match process can thus be very expensive computationally. The second spatial-based technique, the hierarchical approach, is generally more efficient than the block-matching technique [58] essentially due to the large computational time required by the block-matching technique compared to the gradient-based approach. It is based on a multiple resolutions approach building a hierarchy of different resolution frames from an input frame. Grey-level structures are subdivided in a space scale but the underlying motions also get refined in a fine-to-coarse manner. The hierarchical technique is employed in this chapter to overcome the large motion limitation of the robust optical flow estimator developed previously in section 4.1 and is implemented by constructing the Laplacian pyramid [112].

Laplacian pyramid-based hierarchical techniques are commonly used in the literature to provide an initial estimate of motion vectors [4]. Arguments for use of the hierarchical estimation techniques [29] have generally focused on issues of computational efficiency. Simple intuition suggests that if large displacements can be computed using low resolution image information, great savings in computation will be achieved. Higher resolution information can then be used to improve the accuracy of displacement estimation by incrementally estimating small displacements. Due to aliasing effect occurring at high spatial frequency pixels in low resolution frames, motion estimation can become biased and hence misleading. Aliasing is the source of false matches in correspondence solutions or local minima in the objective function. Therefore digital image resolution restricts the maximum estimable motion magnitude.

4.2.2 Laplacian pyramid

A Laplacian pyramid creates a sequence of P images of varying resolutions: $I_t^{(p)}$ with $p = [0 : P-1]$ from an original image as shown in figure 4.7. The left pyramid in this figure contains 4 levels ($P = 4$). The bottom level contains the original and largest frame $I_t^{(0)}$ and the top level frame $I_t^{(P-1)}$ is the smallest frame.

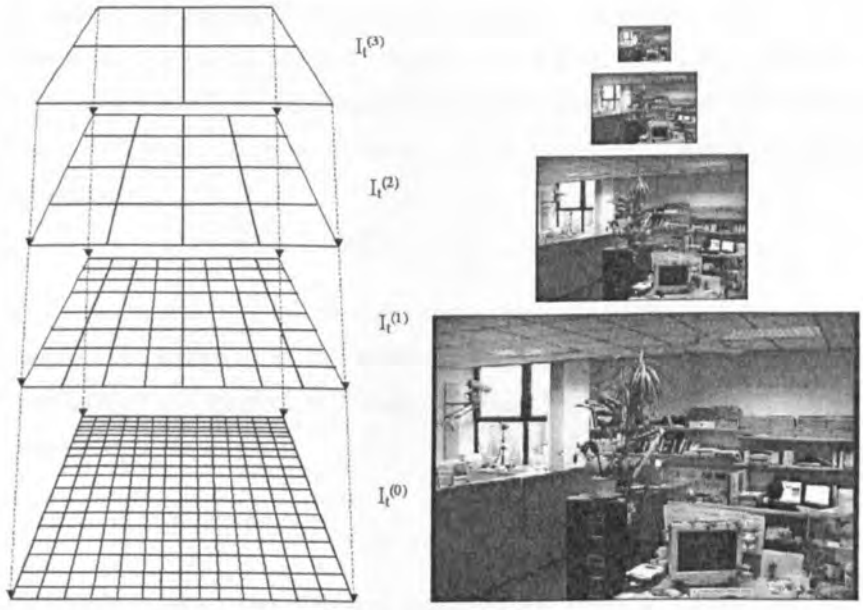

Figure 4.7: Left: structure of a Laplacian pyramid of 4 levels and right: an example

The frame at a particular level in the pyramid has half the dimension (half width and half height) of the frame situated in the above level. Hence, the width and height of the frame at the p^{th} level, denoted $W^{(p)}$ and $H^{(p)}$ respectively can be calculated from the bottom frame dimension as follows

$$W^{(p)} = \frac{W^{(0)}}{2^p}, \text{ and } H^{(p)} = \frac{H^{(0)}}{2^p} \tag{4.48}$$

The most common way to create the sub-images of the pyramid given an image (the bottom level frame) is performed as follows. The grey-level average given by quadruples of neighbouring pixels in an image at a particular level determines the grey-level of the corresponding pixel in the upper level frame:

$$I_t^{(p+1)}\left(\mathbf{x}^{(p+1)}\right) = \frac{1}{4}\sum_{\mathbf{x}'\in\mathcal{W}} I_t^{(p)}\left(\mathbf{x}^{(p)}+\mathbf{x}'\right) \tag{4.49}$$

where $\mathbf{x}'$ is the index of the neighbouring pixels location and

$$\mathbf{x}^{(p)} = 2\,\mathbf{x}^{(p+1)} \tag{4.50}$$

$$\mathcal{W} = \{x=[0:1], y=[0:1]\} \tag{4.51}$$

with $\mathbf{x}^{(p)} = [x^{(p)}, y^{(p)}]^T$ the pixel location in the p^{th} level. Each frame has for origin of their coordinate system their top left frame corner. In this study, the averaging of 4 grey-levels in the previous equation is replaced by the convolution of a Gaussian smoothing kernel of width $\sigma = 0.5$ preserving more local information than the averaging function. This process of averaging followed by sub-sampling is repeated until the lowest resolution frame is obtained.

4.2.3 Hierarchical motion estimation

The robust optical flow estimator developed in section 4.1 is shown to estimate accurately motions with small magnitudes between two successive frames I_t and I_{t+1}. This estimator is implemented in the hierarchy of the Laplacian pyramid described above. Optical flow $\mathbf{a}^{(p)}$ is estimated at the p^{th} level of a pyramid from an initial estimate $\mathbf{a}_0^{(p)}$ and an iteration of small motion updates $\Delta\mathbf{a}^{(p)}$:

$$\mathbf{a}^{(p)} = \mathbf{a}_0^{(p)} + \sum_{k=[0:K-1]} \Delta\mathbf{a}_{k+1}^{(p)} \tag{4.52}$$

The motion at the top level is initialised with a zero motion field where the frame and the motion are the smallest. The frames in all the remaining levels are initialised with a motion updated from the upper frame motion results. Because the ratio between two successive levels is of 2, the ratio between motions is also 2:

$$\mathbf{a}_0^{(p)}(\mathbf{x}^{(p)}) = \begin{cases} 0 & \text{if } p = P-1 \\ 2\,\mathbf{a}^{(p-1)}(\mathbf{x}^{(p-1)}) & \text{else} \end{cases} \tag{4.53}$$

with $\mathbf{x}^{(p)}$ defined in equation 4.50. The final motion estimate $\mathbf{a} = \mathbf{a}^{(0)}$ is evaluated for different levels in the pyramid with different applied motions through the next sections.

4.2.4 Experimental procedures

The procedure developed in chapter 3 is also used here to evaluate the hierarchical optical flow estimator described previously in section 4.2.3. The first section presents four datasets and the ground-truth data. The second section describes how the estimator's performance is evaluated via the percentage of accurately estimated motions over the entire set of pixels within a frame. The evaluation results are displayed in section 4.2.5.

Input datasets and ground-truth motion fields

Ground-truth motion vectors $\Delta\mathbf{x}^g$ are chosen with different magnitudes and construct frames from reference frames by the use of motion compensation and bilinear interpolation resumed in the following equation:

$$I_{t+1}(\mathbf{x}) = I_t(\mathbf{x} + \Delta\mathbf{x}^g) \tag{4.54}$$

The four input datasets are displayed in figure 4.8 and 4.9. They are each composed of 2 frames I_t and I_{t+1} where all pixels undergo the same ground-truth motion vector $\Delta\mathbf{x}^g$.

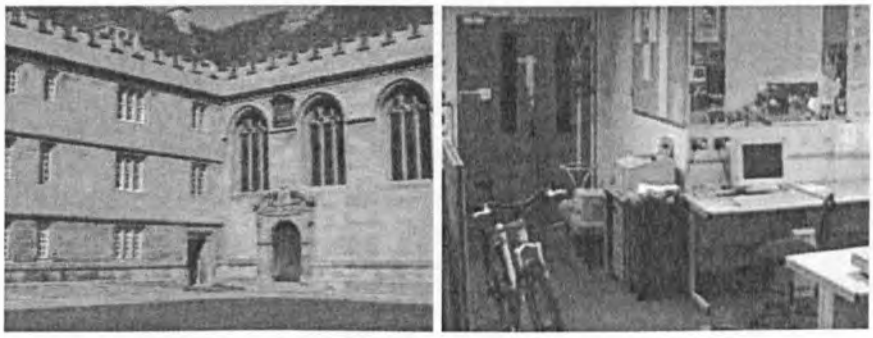

Figure 4.8: Left: 'college' and right: 'lab' input frame

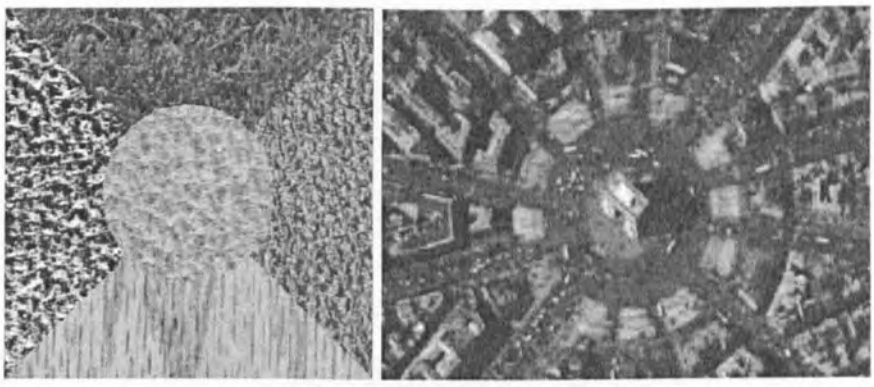

Figure 4.9: Left: 'texture' and right: 'triomph' input frame

Percentage accuracy

The performance of the motion estimator is measured by calculating the percentage P_d of accurately estimated optical flow vectors $\Delta\mathbf{x}$ over the entire set of pixels belonging to the frame $\mathcal{F}$, as follows

$$P_d = \frac{1}{\text{Card}(\mathcal{F})} \sum_{\mathbf{x}_i \in \mathcal{F}} \begin{cases} 1 & \text{if } d(\mathbf{x}_i) < \mathcal{T}_d \\ 0 & \text{else} \end{cases} \quad \text{x100} \tag{4.55}$$

with a threshold $\mathcal{T}_d$ arbitrarily chosen to 0.1 and where $\text{Card}(\mathcal{F})$ is the total number of pixels within the frame $\mathcal{F}$ and the motion error d is defined as the magnitude differences d between

the estimated vector $\Delta\mathbf{x}$ and the corresponding ground-truth vector $\Delta\mathbf{x}^g$:

$$d(\mathbf{x}) = \|\Delta\mathbf{x}(\mathbf{x}) - \Delta\mathbf{x}^g\| \tag{4.56}$$

4.2.5 Evaluation of the Laplacian hierarchical estimator

The hierarchical robust motion estimator described in section 4.2.3 is evaluated in this section. Its performance is evaluated via the percentage P_d of pixels in the frame which are accurately estimated (see section 4.2.4). The percentage P_d are displayed in figures 4.10 for the 'lab' sequence against increasing number of iterations for an applied motion of 5 pixels estimated by the multi-resolution estimator containing 1, 2 and 3 levels in the pyramid. This figure shows that when only one level is used in the pyramid, which corresponds to a non-hierarchical estimation, the system requires at least 15 iterations before converging to a final estimate. On another hand, only a few iterations are necessary for hierarchical systems containing a minimum of 2 levels in their pyramid to converge. Moreover, the more levels in the pyramid the higher the percentage accuracy. Convergence is measured as the K^{th} iteration at which equation 4.57 is valid.

$$\frac{P_d(K+1) - P_d(K)}{P_d(K+1)} < \mathcal{T}_p \tag{4.57}$$

where $P_d(k)$ is the percentage at the k^{th} iteration and the threshold $\mathcal{T}_p$ is arbitrarilly set to 0.01. The convergence K is reported in table 4.2 with the corresponding percentage accuracy $P_d(K)$ for a motion of 5 pixels in magnitude.

levels	K	$P_d(K)$
1	15	52.3
2	2	88.7
3	1	98.8

Table 4.2: Convergence K and respective percentage accuracy $P_d(K)$ when the motion has a magnitude of 5 pixels

The same experiments are performed for various motions whose percentage accuracy is plotted in figure 4.11 for optimal number of iterations of 15, 5 and 5 for 1, 2 and 3 levels in the pyramid respectively. The results of this graph show that the percentage accuracy P_d is greater than 90% when pixel motion is less than approximately 2.5, 5 and 10.5 pixels in magnitude for 1, 2 and 3 levels in the pyramid respectively.

The results with 3 levels in the pyramid are displayed in figure 4.12 for the remaining sequences: 'college', 'lab', 'texture' and 'triomph' sequences. The performance of the optical flow estimator remains similar for the four different datasets and 90% of the pixel motions are accurately estimated if the motion does not exceed 10, 11, 8 and 11 pixels in magnitude for the 'college', 'lab', 'texture' and 'triomph' sequences respectively.

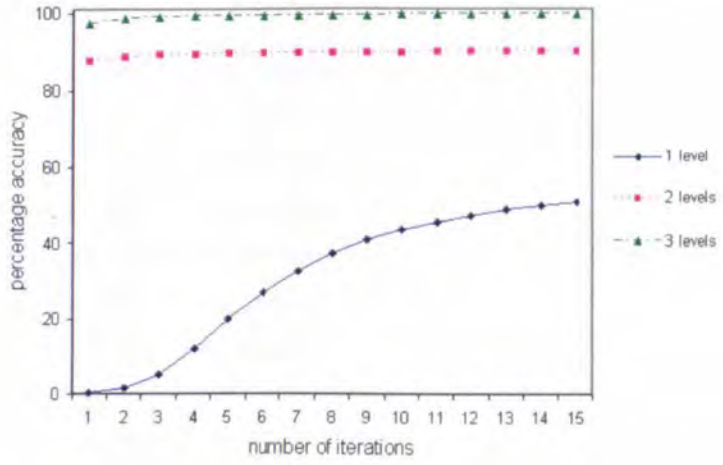

Figure 4.10: Percentage accuracy for a motion of 5 pixels with the 'lab' sequence

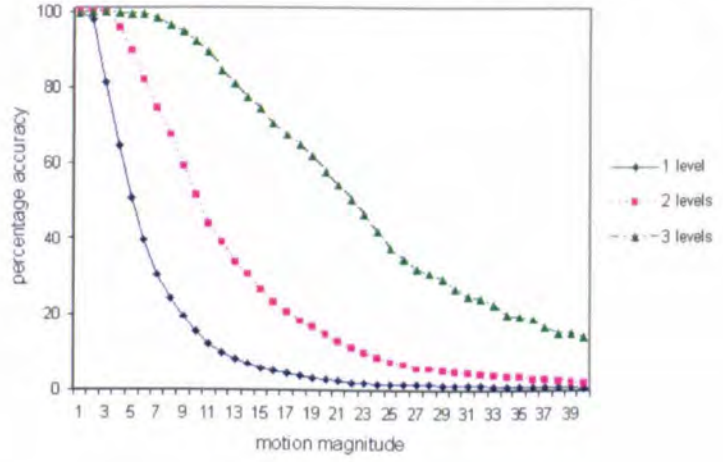

Figure 4.11: Percentage accuracy for different applied motions with the 'lab' sequence

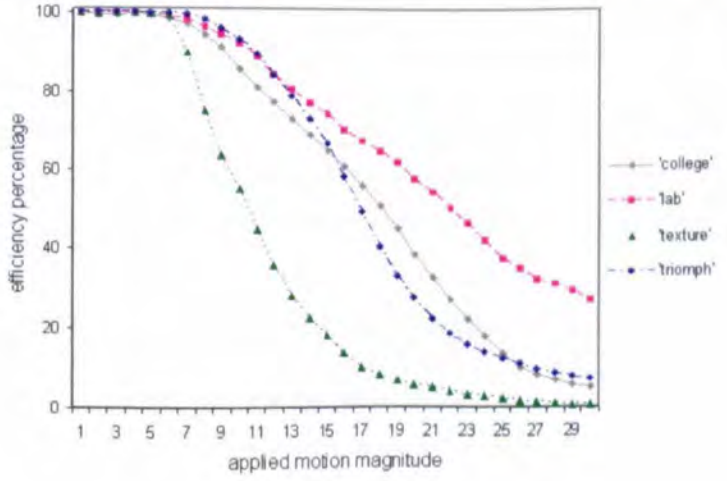

Figure 4.12: 3 pyramid levels results

4.2.6 Conclusion

The motion estimator developed in chapter 3 can estimate accurately more than 90% of the optical flows in a frame if motion magnitudes are not greater than approximately 2 to 3 pixels. However, since this optical flow estimator is developed around the small motion assumption, its performance decreases greatly for larger motions unless initial estimates are provided. A multi-resolution technique is developed in section 4.2.2 which uses the Laplacian pyramid to construct a hierarchy of decreasing resolution frames. This technique first estimates small motions at the coarsest level in the pyramid and larger motions can be then estimated in finer levels using the previously updated motions. The performance of the hierarchical estimator is evaluated in section 4.2.5 where at least 90% of the pixel motions of 4 different datasets with approximately maximum 10 pixels in magnitude can be accurately estimated if the pyramid contains 3 levels (2 sub-levels estimation using non-zero initial estimate). In addition, it was observed that motion estimation initialised with non-zero motion estimate only requires a few iterations to converge to a final estimate whereas the non-hierarchical estimation requires about 15 iterations so that small motions are accurately estimated.

In theory, greater motion magnitudes can be accurately estimated with more levels in the pyramid. However, as the image resolution decreases, aliasing is likely to occur in the optical flow estimation of low-resolution images. Moreover, optical flow requires at least 100 neighbouring pixels and such neighbourhoods are likely to contain several motions in such low-resolution images. It was shown in section 4.1 that motion outliers are not easily eliminated. Consequently multi-resolution tends to introduce a motion smoothing effect across motion boundaries. In the previous experiments, pyramids are constructed so the lowest resolution frame contains at least 50 pixels in width and height from an original size between 300 to 400 pixels in lateral dimension: hence a maximum of 3 or 4 levels are allowed in the pyramids.

The robust and hierarchical estimation described in this chapter is shown to be accurate for a large range of motions. Unlike the multiple motion problem, white noise is successfully eliminated from the neighbourhoods of pixels. Motion estimation is also inaccurate in low-contrasted grey-level regions where there are not enough constraints to estimate any motion. Many applications, involving for example depth retrieval or motion segmentation would make better use of optical flow if it was provided with confidence measurements. The next section attempts to estimate such confidence by the use of covariance expression of the available motion.

4.3 Uncertainty of optical flow measurements

Dense fields of motion vectors describe the image motion from one frame to another. Dense optical flow vectors may be used for example to perform motion detection [70], object segmentation and tracking [8, 135], or it may be used for motion-compensation applications [41, 43], motion study of oceanographic and atmospheric image sequences [37], stereo disparity measure-

ment [15, 125] or depth retrieval from motion segmentation [1]. However, the gradient-based technique described in the previous part of chapter 4 estimates the motion fields accurately only if neighbourhoods of pixels constraining the estimation do not contain multiple motions (as occurs in occlusions around object boundaries), if the dominant motion is not too large, and if there is sufficient grey-level texture. This section focuses on providing extra information about the accuracy of the motion estimates. Section 4.3.1 shows that motion estimation can be expressed in a linear form giving rise to a covariance expression which can be interpreted in terms of uncertainty. Uncertainty can be then geometrically represented by ellipses of confidence representing the degree of accuracy.

4.3.1 Covariance of the motion parameters

The motion parameter updated between iterations, $\Delta\mathbf{a}_{k+1}$ is expressed in chapter 3 in a linear form (see equation 3.78), as

$$\Delta\mathbf{a}_{k+1} = (\Psi_k^T\Psi_k)^{-1}\Psi_k^T\Delta\mathbf{I}_k \tag{4.58}$$

where Ψ is essentially a matrix of grey-level derivatives φ and $\Delta\mathbf{I}$ contains motion compensated grey-level differences ΔI:

$$\Psi_k = [\varphi_{\tau_1}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\varphi_{\tau_1}(\mathbf{x}'_n,\mathbf{a}_k),\ldots,\varphi_{\tau_m}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\varphi_{\tau_m}(\mathbf{x}'_n,\mathbf{a}_k)]^T \tag{4.59}$$

$$\Delta\mathbf{I}_k = [\Delta I_t^{\tau_1}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\Delta I_t^{\tau_1}(\mathbf{x}'_n,\mathbf{a}_k),\ldots,\Delta I_t^{\tau_m}(\mathbf{x}'_1,\mathbf{a}_k),\ldots,\Delta I_t^{\tau_m}(\mathbf{x}'_n,\mathbf{a}_k)]^T \tag{4.60}$$

$$\varphi_\tau(\mathbf{x},\mathbf{a}_k) \approx \Delta t\nabla I_t(\mathbf{x})X(\mathbf{x}) \tag{4.61}$$

$$\Delta I_t^\tau(\mathbf{x},\mathbf{a}_k) = I_t\mathbf{x} - I_\tau(\mathbf{x}+\Delta tX(\mathbf{x})\mathbf{a}_k) \tag{4.62}$$

The terms of the previous equation are derived from the pixels within the spatio-temporal window of axis $\mathbf{x}'$ and τ: the constraining neighbourhood of pixels (see section 3.3). $X(\mathbf{x})$ in equation 4.62 is the matrix of the planar motion model (see section 3.2). Due to the linearity of equation 4.58, the covariance of the motion $\Delta\mathbf{a}_{k+1}$ can be expressed as follows

$$\Lambda_{\Delta\mathbf{a}_{k+1}} = \left((\Psi_k^T\Psi_k)^{-1}\Psi_k^T\right)\Lambda_{\Delta\mathbf{I}_k}\left((\Psi_k^T\Psi_k)^{-1}\Psi_k^T\right)^T \tag{4.63}$$

$$= (\Psi_k^T\Psi_k)^{-1}\Psi_k^T\Lambda_{\Delta\mathbf{I}_k}\Psi_k(\Psi_k^T\Psi_k)^{-1} \tag{4.64}$$

where $\Lambda_{\Delta\mathbf{I}_k}$ is the covariance of the vector of N grey-level differences after motion compensation: ΔI. Each of these are assumed to be represented by uncorrelated white noises with variance σ_i^2 where $i = [1:N]$ as represented in equation 4.65. In addition it is also assumed that all these noise sources can be modelled by a single noise of variance $\sigma_{\Delta I}^2$ at each of the pixels in the neighbourhood. Thus, the covariance of $\Delta\mathbf{I}_k$ can be approximated by equation 4.67 where I is the identity matrix of size NxN:

$$\Lambda_{\Delta\mathbf{I}_k} \approx \begin{bmatrix} \sigma_1^2\ldots 0 \\ \vdots\ddots\vdots \\ 0\ldots\sigma_N^2 \end{bmatrix} \tag{4.65}$$

$$\approx \sigma_{\Delta I}^2 \begin{bmatrix} 1 \dots 0 \\ \vdots \ddots \vdots \\ 0 \dots 1 \end{bmatrix} \tag{4.66}$$

$$\approx \sigma_{\Delta I}^2 I \tag{4.67}$$

Using this new equation for $\Lambda_{\Delta \mathbf{I}_k}$ in the optical flow covariance equation 4.64 gives:

$$\Lambda_{\Delta \mathbf{a}_{k+1}} \approx \sigma_{\Delta I}^2 (\Psi_k^T \Psi_k)^{-1} \Psi_k^T \Psi_k (\Psi_k^T \Psi_k)^{-1} \tag{4.68}$$

$$\approx \sigma_{\Delta I}^2 (\Psi_k^T \Psi_k)^{-1} \tag{4.69}$$

Motion displacement at a pixel is calculated from the estimated motion model by the following equation:

$$\Delta \mathbf{x} = \Delta t X(\mathbf{x}) \mathbf{a} \tag{4.70}$$

Estimating the covariance of the vector $\Delta \mathbf{x}$ involves the covariance estimation of $\mathbf{a}$ (the sum of motion estimates between iterations described in equation 3.40), *i.e.*

$$\mathbf{a} = \mathbf{a}_0 + \sum_{k=0}^{K-1} \Delta \mathbf{a}_{k+1} \tag{4.71}$$

However, unlike equation 4.58, the final motion of equation 4.71 (and hence the 2D vector $\Delta \mathbf{x}$) is the sum of K iterative updates, resulting in a complex covariance expression. In this study, it is considered that the motion uncertainty $\Lambda_{\mathbf{a}}$ is approximately K times the covariance of the estimated motion update $\Delta \mathbf{a}_K$ at the last iteration. Using equation 4.70 with the covariance of the final motion update in equation 4.69 uncertainty can be approximated by

$$\Lambda_{\Delta \mathbf{x}} \approx K \left(\Delta \tau X(\mathbf{x})\right) \Lambda \mathbf{a} \left(\Delta \tau X(\mathbf{x})\right)^T \tag{4.72}$$

$$\approx \Delta t^2 \sigma_{\Delta I}^2 X(\mathbf{x}) (\Psi_K^T \Psi_K)^{-1} X(\mathbf{x})^T \tag{4.73}$$

where $\Psi = \Psi_K$ is a vector of N row matrices φ (see equation 4.59) which contains p derivative terms ∇I_t. Developing equation 4.73 gives

$$\Lambda_{\Delta \mathbf{x}} \approx K \Delta t \, \sigma_{\Delta I}^2 X(\mathbf{x}) \left(\sum_{\tau} \sum_{\mathbf{x}'} \varphi_\tau^T(\mathbf{x}', \mathbf{a}) \varphi_\tau(\mathbf{x}', \mathbf{a}) \right)^{-1} X(\mathbf{x})^T \tag{4.74}$$

$$\approx K \Delta t \, \sigma_{\Delta I}^2 X(\mathbf{x}) \left(\sum_{\tau} \sum_{\mathbf{x}'} \Delta \tau^2 X^T(\mathbf{x}') \nabla I_t^T(\mathbf{x}') \nabla I_t(\mathbf{x}') X(\mathbf{x}') \right)^{-1} X(\mathbf{x})^T \tag{4.75}$$

In addition, if optical flow is computed between only two frames with Δt normalised to 1 and a translational motion model is fitted to the neighbourhood of pixels ($X = I$) then the covariance simplifies to:

$$\Lambda_{\Delta \mathbf{x}} \approx K \sigma_{\Delta I}^2 M^{-1} \tag{4.76}$$

where

$$M^{-1} = \left(\sum_{\mathbf{x}'} \nabla I_t^T(\mathbf{x}') \nabla I_t(\mathbf{x}') \right)^{-1} \tag{4.77}$$

$$= \begin{bmatrix} \sum_{\mathbf{x}'} I_x^2(\mathbf{x}') & \sum_{\mathbf{x}'} I_x(\mathbf{x}')I_y(\mathbf{x}') \\ \sum_{\mathbf{x}'} I_x(\mathbf{x}')I_y(\mathbf{x}') & \sum_{\mathbf{x}'} I_y^2(\mathbf{x}') \end{bmatrix}^{-1} \tag{4.78}$$

$$= \frac{1}{D} \begin{bmatrix} \sum_{\mathbf{x}'} I_y^2(\mathbf{x}') & -\sum_{\mathbf{x}'} I_x(\mathbf{x}')I_y(\mathbf{x}') \\ -\sum_{\mathbf{x}'} I_x(\mathbf{x}')I_y(\mathbf{x}') & \sum_{\mathbf{x}'} I_x^2(\mathbf{x}') \end{bmatrix} \tag{4.79}$$

and

$$I_x(\mathbf{x}') = \frac{\partial I_t(\mathbf{x}')}{\partial x} \tag{4.80}$$

$$I_y(\mathbf{x}') = \frac{\partial I_t(\mathbf{x}')}{\partial y} \tag{4.81}$$

$$D = \sum_{\mathbf{x}'} I_x^2(\mathbf{x}') \sum_{\mathbf{x}'} I_y^2(\mathbf{x}') - \left(\sum_{\mathbf{x}'} I_x(\mathbf{x}')I_y(\mathbf{x}') \right)^2 \tag{4.82}$$

Therefore, according to equation 4.76 with equation 4.79, optical flow covariance and hence its uncertainty are directly dependent on the inverse of the Hessian matrix (the spatial grey-level derivatives) scaled by the variance of the grey-level noise present after motion compensation. Thus, the spatial distribution of the sum of derivatives in both vertical and horizontal directions influences motion estimation, and the larger the motion compensated errors the larger the uncertainty. Moreover, in the above equations, if the term D is very small the covariance matrix becomes very large, and hence the uncertainty of the motion estimate. This term D (equation 4.82) is by definition the determinant of the Hessian matrix. Therefore if this term is zero the Hessian matrix is singular and the inverse matrix operation is not possible. For example, $D = 0$ for regions with uniform gradients (represented for example by setting I_x to aI_y in equation 4.82 for all neighbouring pixels).

Robust case

Section 4.1 developed a robust estimation procedure for optical flow that differs only by a weighting operation within the matrix Ψ containing the derivatives terms contributed by each of the neighbourhood pixels:

$$\Delta \mathbf{a}_{k+1} = (\Psi_k'^T \Psi_k)^{-1} \Psi_k'^T \Delta \mathbf{I}_k \tag{4.83}$$

where Ψ' is the weighted Ψ (see section 4.1.2). Using the previous linear approach with equation 4.83, it can easily be shown that the covariance of the motion vector $\Delta\mathbf{x}$ is approximately

$$\Lambda_{\Delta\mathbf{x}} \approx K\sigma_{\Delta I}^2 \left(\Psi_k'^T \Psi_k\right)^{-1} \Psi_k'^T \Psi_k' \left((\Psi_k'^T \Psi_k)^T \right)^{-1} \tag{4.84}$$

$$\approx K\sigma_{\Delta I}^2 \sum_{\mathbf{x}'} \left(w(e_\tau(\mathbf{x}', \mathbf{a})) \nabla I_t^T(\mathbf{x}) \nabla I_t(\mathbf{x}) \right)^{-1} \left(w(e_\tau(\mathbf{x}', \mathbf{a})) \nabla I_t^T(\mathbf{x}) \nabla I_t(\mathbf{x}) \right) \tag{4.85}$$

$$\left(\left(w(e_\tau(\mathbf{x}', \mathbf{a})) \nabla I_t^T(\mathbf{x}) \nabla I_t(\mathbf{x}) \right)^T \right)^{-1} \tag{4.86}$$

$$\approx K\sigma_{\Delta I}^2 \sum_{\mathbf{x}'} \frac{1}{w^2(e_\tau(\mathbf{x}', \mathbf{a}))} \left(\nabla I_t^T(\mathbf{x}) \nabla I_t(\mathbf{x}) \right)^{-1} \tag{4.87}$$

where $w(e)$ is the weighted associated with the motion compensated grey-level error e and takes a maximum value of 1 for inlier errors and tends to 0 for large outliers. Comparing equation

4.87 with equation 4.76, the covariance of the robust estimator is lower than the covariance of the least-squares estimator when outliers are present in the neighbourhood of pixels. Outliers are discarded from the covariance estimation and hence the uncertainty is not affected by them when robust estimation is performed.

4.3.2 Representing covariance and confidence

The optical flow covariance matrix expressed in the previous section can be geometrically represented as defined in the first section and a way of displaying confidence of the estimated motion vectors is described in the second section.

Visualising covariance

The Gaussian uncertainty $\Lambda_{\Delta\mathbf{x}}$ can be visualised by drawing an ellipse containing some percentage of the distribution. The following equation

$$\mathbf{x}^T \Lambda_{\Delta\mathbf{x}}^{-1} \mathbf{x} = c^2 \tag{4.88}$$

where $\mathbf{x}$ is a local coordinate system centred at the end of the optical flow vector (*i.e.* $\mathbf{x} + \Delta\mathbf{x}$ on the image plane) and c^2 is a Mahalanobis distance between a point in $\mathbf{x}$ and the mean. Since $\Lambda_{\Delta\mathbf{x}}$ represents a bi-variate Gaussian distribution, then $c^2 = 2.48$ represents the 95% boundary of a Chi-squared distribution. This distribution can be geometrically represented by for example ellipses [38, 23] or rectangles as displayed in figure 4.17.

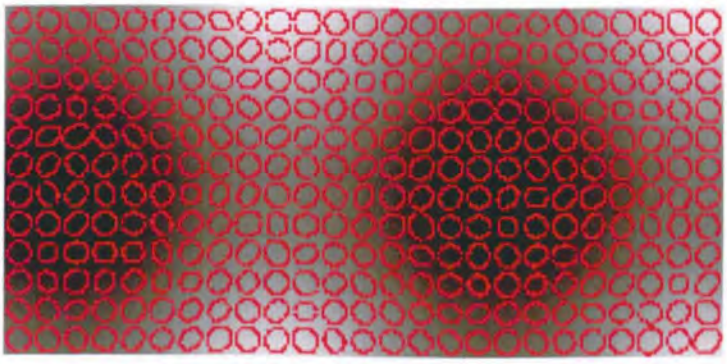

Figure 4.13: Visualising covariances through ellipsoids

As will be discussed in section 4.3.3, the shape of these ellipses (and hence the structure of the covariances) show a distinct correlation with local grey-level structures.

Visualising confidence

The greater the area of the ellipse defined in equation 4.88 which represents the optical flow covariance, the greater the uncertainty associated with the optical flow. On another hand, the

smaller the covariance, the more confident the estimation. A confidence measurement is defined as the inverse of the ellipse area A_e.

where $det(\Lambda)$ is the determinant of the matrix Λ. It should be noted that the confidence measurement developed in this section reflects the consistency of the constant brightness assumption. However, motion of pixels in low textured regions which lacks grey-level structure are likely to be inaccurately estimated with high confidence. Therefore, confidence does not reflect the uncertainty of the estimated motion in such low-contrasted regions.

4.3.3 Investigating the uncertainty of optical flow estimators

Random Gaussian noise is introduced into two identical sine frames displayed in figure 4.14. The motion field is thus zero between these two frames. However, non-zero optical flow results will be obtained when random noise is introduced.

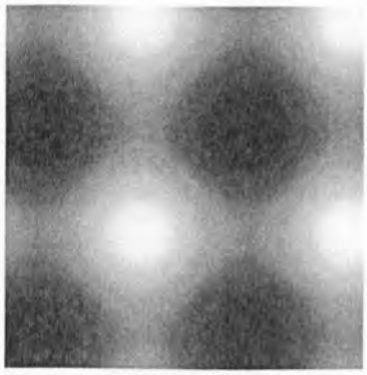

Figure 4.14: 'big-sine' frame example

How does grey-level structure affect optical flow estimation?

The angle difference between the direction of the measured optical flow vector and the direction of the gradient vector will be uniformly distributed if the local grey-level structure does not impact on the estimation of optical flow. Histograms of the angular differences are displayed in figure 4.15 for a small noise level of 1 and a high level of 10. For low noise levels, the majority of the optical flow vectors are located in the range $[-45 : 45]$ degrees from the gradient's direction. However these angle differences are approximately uniformly distributed over the entire range $[-90 : 90]$ degrees for the higher levels of noise. Hence, optical flow vectors in noiseless images are highly correlated to the direction of the gradient and reflects the aperture problem introduced in section 3. However, when the introduced level of noise becomes large, the optical flow becomes uncorrelated with the gradient direction.

This biasing effect has also been noted in the work of Fermuller *et al* [49]. The relationship between the ground-truth flow $\Delta\mathbf{x}^g$ and estimated flow $\Delta\mathbf{x}$ was shown to be given by

$$\Delta\mathbf{x} \approx \Delta\mathbf{x}^g \; - \; m \; \sigma_{\nabla I}^2 M^{-1} \Delta\mathbf{x}^g \tag{4.89}$$

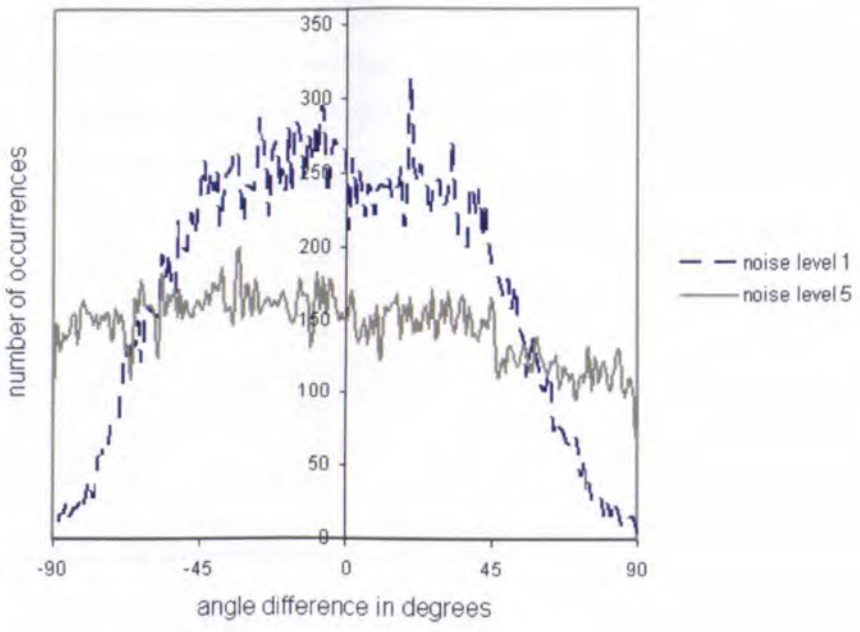

Figure 4.15: Histogram of angle differences between the direction of the estimated optical flow vector and the direction of the gradient vector

where M, previously defined in equation 4.77, contains the m spatial derivatives over a small neighbourhood which pixels are located at $\mathbf{x}' = [\mathbf{x}'_1 : \mathbf{x}'_m]$. It is assumed an equal variance $\sigma^2_{\nabla I}$ for the zero-mean noise in the spatial derivatives in the x and y directions. From equation 4.89, the flow is underestimated in length and its orientation is biased towards the majority of the gradients, as illustrated by figure 4.15.

How does grey-level structures affect covariance of motion estimation?

Having explored the bias in optical flow estimates, the effect of structure on motion uncertainty is explored. Figure 4.16 plots the histogram of differences in angle between the direction of the ellipse's minor axis (representing the normal to the expected direction of the optical flow) and the direction of the gradient at each pixel in the frame. The level of introduced noise is represented by the width of the Gaussian kernel. Figure 4.16 shows that the angle differences belong to a symmetric unimodal normal-like distribution with a mean of zero degree and variance which increases with the additive grey-level noise level. Therefore, when there is little noise, the expected optical flow lies in an ellipse area with major direction normal to the direction of the gradient. On another hand, when the level of introduced noise increases, the ellipse areas are less correlated with the direction of the grey-level gradient. This is demonstrated visually in figure 4.17 which shows optical flow errors covariance for two examples with noise levels of 1 and 5 respectively. They are smaller in low noise levels and they are less correlated with the main direction of the gradient when the noise increases. Grey-level structure, therefore, has a

significant effect on the optical flow covariance. This is the aperture problem in that the least certain direction of the optical flow lies along a direction normal to the grey-level gradient, and is pronounced in neighbourhoods which lacks grey-level structure.

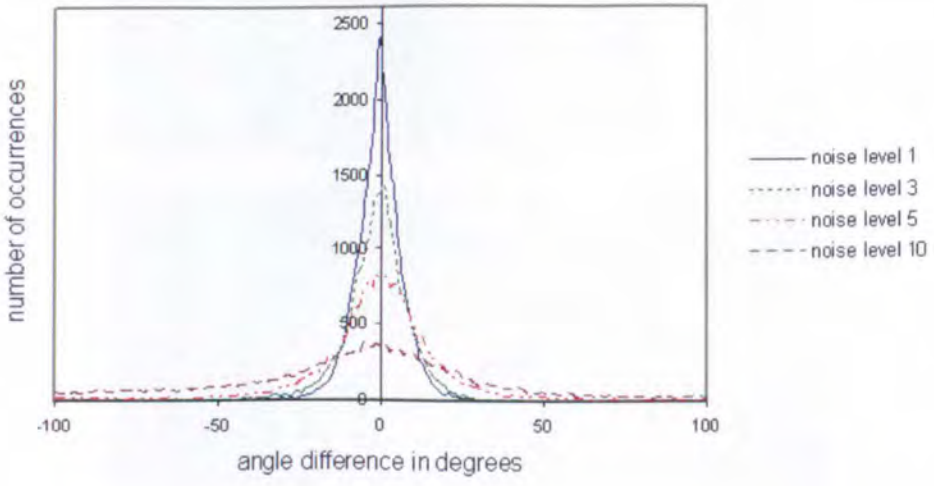

Figure 4.16: Histogram of angle differences between the direction of the covariance and the direction of the gradient vector from the 'sine' data

Displaying confidence measurements

Confidence was expressed in section 4.3.2 as the inverse of the area of the ellipse representing the optical flow covariance matrix. The greater the area, the greater the uncertainty associated with the optical flow and vice versa. Figure 4.19 presents the confidence values calculated for the motion estimated between the two real captured frames shown in 4.18. Confidences are displayed on the left of figure 4.19 for different estimators (where low grey-levels areas correspond to low confidence areas and vice versa). On the right hand side of figure 4.19, the corresponding 2D estimated flow vectors are superimposed on top of the original input frame for each of the different motion estimators. These results show that, in general, the use of robust statistics and hierarchy increases the confidence of the results. Motion remains corrupted by the presence of multiple motions at object boundaries. However, this problem is signalled by the low confidence obtained in these regions.

4.3.4 Summary

It has been shown that optical flow and the associated confidence are dependent on the local distribution of the spatial gradients. The covariance matrices can be represented geometrically by ellipses of confidence allowing the user to visualise the areas of high confidence. Optical flow results are shown to be accurate in textured areas with high confidence when a robust and hierarchical scheme is adopted but looses accuracy and confidence when occlusions occur. A

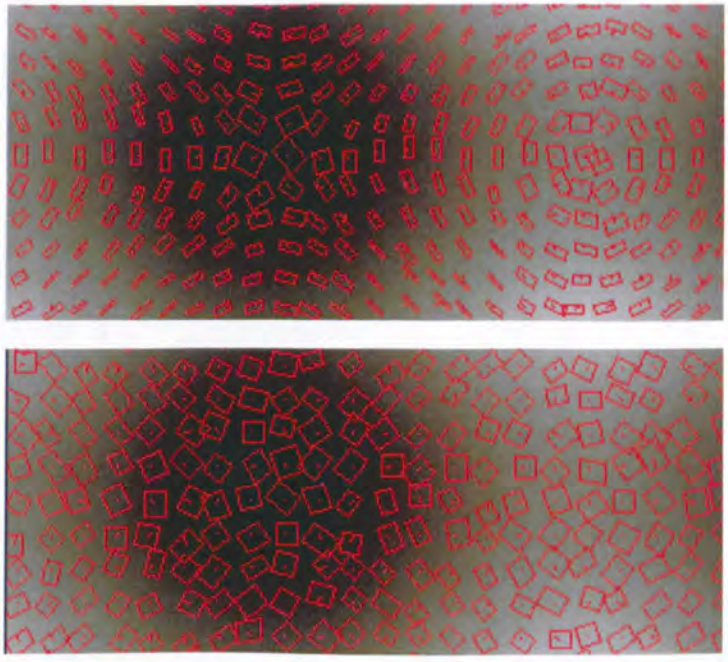

Figure 4.17: Flows and ellipses bounding boxes examples of the 'sine' sequence for a noise intensity of top: 1 and bottom: 5

method for providing a confidence measurement is developed based on how much the optical flow results respect the constant brightness assumption. More specifically, if the pixel neighbourhood generates low motion-compensated errors after motion estimation, then the final confidence is expected to be high and vice versa.

4.4 Note on computational time

Non-robust motion estimation is dominated by the time required to build optical flow matrices - denoted T_{build} in section 3.7. T_{build} can be modelled as

$$\mathrm{T}_{\text{build}} \approx N(a_1 p^2 + a_2 p + a_3) \tag{4.90}$$

where N is the number of constraining pixels in a neighbourhood, p is the number of motion parameters and a_1, a_2 and a_3 are time constants. The weighting operations involved in the robust estimation can be modelled by

$$\mathrm{T}_{\text{robust}} \approx N a_4 \tag{4.91}$$

where a_4 is a time constant. Calculation of the weights associated with the motion compensated errors given at each neighbouring pixels is possible if the median of the distribution of errors is known. A sorting operation is usually involved in calculating a median but in our implementation is replaced by a simple and quick histogram. Using a 850MHz PC Pentium III, the timing

Figure 4.18: 'garden' frame example captured at left: t and right: $t+1$ with unique translational ground-truth motion vector applied for all pixels

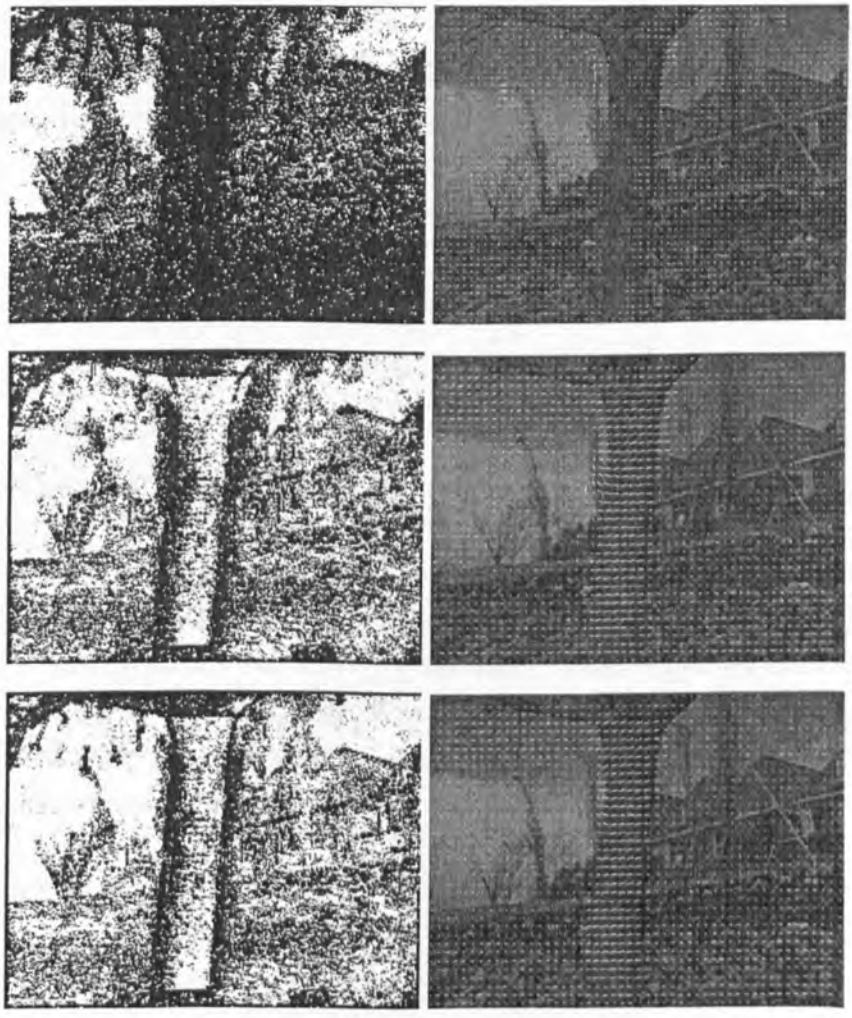

Figure 4.19: Left: Confidence map and right: flows results with on top the non-hierarchical/non-robust result, in the middle the hierarchical/non-robust results and in the bottom frames the hierarchical/robust results

parameters are estimated as

$$[a_1,\ a_2,\ a_3] \approx [5.87\text{x}10^{-8},\ 2.91\text{x}10^{-7},\ 1.58\text{x}10^{-6}] \text{ seconds} \quad (4.92)$$

$$a_4 \approx 1.95\text{x}10^{-6} \text{ seconds} \quad (4.93)$$

It was estimated that the time required to compute the weights becomes greater than the time to build the matrices for motion model with a very large number of motion parameters compared to the eight parameters of the planar model. Therefore the time to compute weights is negligible in the overall optical flow estimation which was estimated to be a $\mathcal{O}(N)$ estimator per pixel in section 3.7.

When the hierarchical motion estimation is deployed to overcome the large motions limitation, extra computational time is necessary. If a Laplacian pyramid (described in section 4.2) is constructed with P levels, then P optical flow estimation on different resolution images are performed. If the pyramid is constructed from a frame $\mathcal{F}$ on its bottom level containing $\text{Card}(\mathcal{F})$ pixels, then sub-levels frames at a level p contains $\text{Card}(\mathcal{F})/4^p$ pixels with $p = [0 : P-1]$. If T is the time required to compute optical flow at all the pixels of a frame, then the overall computational time to compute optical flow of all pixels of an infinite number of levels in a pyramid is a geometric series as follows

$$T + \frac{T}{4} + \frac{T}{16} + \frac{T}{64} + \ldots = T\left(1 + \sum_{p=1}^{\infty}\left(\frac{1}{4}\right)^p\right) \quad (4.94)$$

$$= T\left(1 + \frac{1/4}{1-1/4}\right) = T + \frac{T}{3} \quad (4.95)$$

Therefore hierarchical estimation increases at most $1/3$ of the time involved in non-hierarchical estimation.

A frame in a pyramid is built by sub-sampling the frame situated above it in the hierarchy smoothed by a Gaussian kernel of width 0.5. This procedure is considered quick compared to the estimation of optical flow and similarly to this latter, the construction of one frame in the pyramid takes four times from one level to a lower level. Hence, if the time to construct the first sub-frame from the original frame takes T_c then an infinite pyramid would take $T_c + T_c/3$ which can be considered negligible compared to the time to compute optical flow *i.e.* $T + T/3$.

4.5 Conclusion

This chapter addresses the three main problems associated with optical flow estimation *i.e.* noise contamination, multiple motions and large motions. Noise was addressed in the first section and successfully eliminated by the use of robust statistics. However, when noise level is too important such as the motion noise caused by multiple motions occurring at object boundaries, robust statistics could not successfully cope with the outliers. Large motions could be recovered in the second section by the use of a multi-resolution technique. The third section addresses

another problem associated with optical flow which is the problem of providing uncertainty measurements of the pixel motions.

Robust statistics

The least-squares regression scheme used to minimise the sum of motion compensated errors derived from the constant brightness equation performs well when there is little noise (see the results of chapter 3). The term outlier was introduced to describe any noise that does not belong to a normal distribution within the neighbourhood centred around the pixel for which optical flow motion is computed. The first component of this chapter investigates the impact of motion boundaries on the estimation of optical flow from pixel neighbourhoods. Results demonstrated that the least-squares technique is non-robust and zero-tolerant to outliers.

Robust statistics was then introduced to address this problem. Estimators were implemented as weighted least-squares estimators. The median estimator was chosen for its robustness, efficiency and computational speed. Such robust techniques seek to weight strongly data points belonging to the dominant motion, and conversely de-weight pixels associated with the least dominant motion(s). Experimental evidence suggests that the median estimator efficiently rejects outliers when they represent at most 30% of the entire distribution. Although the median estimator is proven to improve greatly the accuracy of the motion estimates compared to the least-squares, it cannot fully reject 100% of the outliers at motion boundaries.

Large motions

Chapter 3 clearly showed that a gradient-based optical flow estimator operates efficiently on a very limited range of motion magnitudes of the order of a few pixels. Over-smoothing the frame with a Gaussian kernel whose width is matched to the motion magnitude is not a practical solution as it is dependent on the unknown motion. A hierarchical technique was introduced using the well known Laplacian to build a coarse-to-fine hierarchy of sub-sampled frames. The top level of the pyramid contains the smallest or coarsest sub-sampled frame and the bottom level contains the original or finest frame. Sub-sampling addresses the problem of large motions as any pixel motion in the original frame also gets progressively sub-sampled up the pyramid. Estimation at the top level of the pyramid is initialised with zero motion estimates and each of the lower levels is initialised with the magnified motion field from the corresponding upper level. It was demonstrated that such an update and projection approach enables the hierarchical estimator to accurately recover larger motion magnitudes as the number of levels in the pyramid increases. For example, a three-level pyramid allows efficient estimation of motion approximately with maximum 10 pixels in magnitude. Experiments also showed that any level of the pyramid initialised with reasonable estimate of the motion field does not require as much iterations to converge. While up to 15 iterations are needed for the top level, typically 5 iterations is sufficient for convergence in any lower level of the Laplacian pyramid. The number of allowed levels in one

pyramid depends on original frame size and also on the density of grey-level structure. (Some structures can vanish with deep sub-sampling).

Motion uncertainty and confidence

A confidence estimation based on the degree to which the motion estimates conforms to the constant brightness equation rule is derived in this chapter. More specifically, neighbourhoods of pixels which give low motion-compensated errors have a high confidence measurement and vice versa. The linear form of the optical flow estimator provides directly a covariance matrix of the displacement vectors in both the least-squares and robust case. The covariance matrices are represented visually by mean of ellipses to allow the user to visually evaluate the uncertainty associated with optical flow results. The confidence results showed that combining hierarchical estimation with robust statistics greatly improves the accuracy and the confidence of the motion estimates. Moreover, the results also showed that the optical flow results are highly reliable in textured areas whereas confidence decreases at motion boundaries or when the image lacks grey-level structures.

Chapter 5

Hyperplane global motion estimator

Global motion estimation is an important task in a variety of image processing applications. The term global motion is commonly used to describe the motion of a background scene in video sequences generally induced by camera motion. Motion in a video sequence is either local due to object movement or global due to camera movement or zoom. The motion due to object movement is referred to as *local motion* or *object motion* as it only relates to the subset of pixels projected from an object.

One key application of global motion estimation is to perform video annotation. Once a video sequence is annotated the user can make queries about motion related events in the video. Moreover, motion is frequently associated with semantic information making it possible to detect certain situations depending on the motion effects, *e.g.* in sports videos. Motion estimation plays an important role in video data compression which exploits the high temporal redundancy between successive frames in a video sequence to achieve high compression efficiency as performed in MPEG-4 or to achieve motion classification as in MPEG-7. It can also be used for segmenting images into objects moving at different speeds for computer vision applications.

The most often employed motion estimation technique in video coding, such as the standardised MPEG-1/2, is one of block-matching, which gives estimates of the combined local and global motion. Since the global motion is generated by camera movement, it can be represented, in theory, by a few parameters. Hence, the separation of global and local motion may lead to simpler and more efficient motion information representation. Also, the global motion components contained in the motion vectors may confuse an unsophisticated motion-based segmentation algorithm in the identification of moving objects. When the global motion components are removed, the remaining local motion information can be more readily used for moving object identification.

Historically, there are two main approaches to motion estimation, namely matching schemes

and schemes based on the optical flow principle. While matching techniques, such as block-matching, rely on the analysis of the image intensity and the minimisation of a certain cost function as a matching criterion, the optical flow principle is based on the analysis of the spatial and temporal gradients in the video signal.

The mathematical complexity of the global motion model directly affects the possible accuracy of the motion description, but also the computational complexity and stability. For instance the planar perspective model with eight parameters is suitable to describe the global motion completely if the assumption of a pin-hole camera model is satisfied and if the projective projection model is respected (see chapter 3.2). More sophisticated formulations include also lens distortion parameters. For many applications the affine model is sufficient which consists of a linear transformation with six parameters.

A new technique called the hyperplane approximation technique addresses global motion estimation and can be applied to computer vision applications. The hyperplane approximation, as studied by Jurie in [77, 76], is a method of learning the relationship between the applied motion parameters and the induced pixel differences between consecutive frames, also known as the DFD (displaced frame difference). Solving for a given motion is equivalent to solving a set of linear equations in a multi-dimensional space *i.e.* the hyperplane equations. This new technique for motion estimation was successfully applied to template matching techniques [77] and appears to avoid the need for extracting the grey-level gradients. Moreover gradient-based techniques such as the optical flow method developed in a previous chapter, require a large number of constraining points and are very sensitive to large motion magnitudes. A hyperplane-based technique is then implemented in this chapter and evaluated against this standard optical flow technique for global motion estimation.

This chapter focuses on developing a global motion estimator for surveillance applications. Background scenes are captured by video cameras without any foreground objects. The background frames are in general formed by distant background structure. The recent existing techniques and their applications are reviewed in section 5.1 showing how authors provide on-line global motion estimation followed by a segmentation process to separate local and global motions. A standard gradient-based optical flow technique and a novel hyperplane estimator are developed in section 5.3 and 5.2 respectively and their performances are compared in section 5.5.

A novel sampling technique based on randomly sampling strong edges [128] is developed in section 5.4 to provide a desired number of edge-based pixels that input of the motion estimators. The details of implementation and the evaluation methodology are also given in section 5.4. The results of the comparable non-iterative gradient-based and hyperplane techniques show that both estimators are not 100% reliable in estimating accurately the global motion in various

surveillance video sequences. The non-iterative gradient-based estimator is found to be dependent on the unknown motion magnitude as observed in chapter 3. The hyperplane estimator appears to avoid the need to have prior knowledge of the largest motion magnitude it is expected to estimate in a sequence. However, as it is computationally expensive, it generally cannot be a frame-rate estimator and also lacks accuracy.

5.1 Review

Global motion is usually performed using either pixel correlation techniques (block-matching-based techniques) or optical flow-based techniques. Global motion estimation is used to detect typical camera operations associated with particular scenarios in sports events. For example, Kokaram and Delacourt [82] propose a gradient-based optical flow approach for estimating global motion. The algorithm works iteratively where segmentation is performed iteratively between local and global motion which then weights the foreground objects according to their motion. The project recognises successfully more than 88% of cricket events and intends to include sound analysis for complete automated recognition and analysis of the game.

Wei *et al* [137] construct an algorithm for segmenting foreground objects from background objects using the combined information present in temporal and spatial derivatives of the frames. The local motion pixels are grouped iteratively until the system converges toward a fixed number of segmented areas. The algorithm relies on two important assumptions: video sequences have typical background-foreground structured scenes where the background is dominant in the sequences and the foreground consists of connected rigid bodies; and background motions are caused only by camera operations.

Heuer and Kaup [65] avoid any high computational costs involved in gradient-based motion estimator and rather use the available rough estimates of translational motion vectors given by the encoded MPEG-4 technique. These motion vectors are estimated using a block-matcher technique. The segmentation between the local and global motion is performed robustly with a 4 parameters global motion model fitted to the motion vectors. Smolic *et al* [119, 118] combine a feature matching technique to find correspondences followed by an optical flow technique to estimate the global motion parameters. A hierarchical strategy is applied for the estimation. First a translation model is fitted followed by an affine model, and finally an higher-order motion parameter model is fitted. The estimator is robust and computationally efficient in segmenting foreground objects from the background scene. The developed algorithm is allowed to build mosaics of the background with great accuracy.

Saez *et al* [111] construct an algorithm which makes use of a feature extraction technique based on the generalised Hough transform, which is able to provide rotation, scale and displacement parameters when comparing two consecutive frames from a video sequence, and hence allow for the pan, tilt, swing (rotation along the z-axis) and zoom effects to be measured. In the earlier work of Dufaux and Moschen [44], mosaicking which consists of registering images onto

a large single image, is performed by first estimating all the local motions and then by applying a parametric motion model to segment the global motions induced by the camera.

Global motion estimation algorithms can also be used to extract the 3D information of foreground and background object. For example Tzovaras *et al* [134] propose in a two-step algorithm for robust global 3D motion estimation. In the first step, a rough 2D global motion estimation method is achieved using block-matching. Using random sampling on the edges of the image, features of interest are selected and their 2D motions allow for an initial estimate of the 3D camera motion. In the second step, the camera movement is modelled by an affine motion model in which parameters are estimated by a robust method allowing for depth retrieval based on disparity measurement. The method works iteratively in order to remove progressively the detected foreground outliers.

Other techniques use frequency analysis to estimate optical flow and to estimate separately local and global motions in video sequences, as performed by Bruno and Pellerin [27, 26]. The hyperplane approximation estimates motion successfully in template matching applications [77] but the estimator is not of great interest in the literature for general global motion estimation applications.

5.2 Hyperplane global motion optical flow

The potential new technique for estimating global motion, the hyperplane estimator, is developed in this section as a global motion estimator before being applied to surveillance applications in the later section 5.5. Hyperplane motion estimation is based on building a set of linear relations between motion perturbations and DFDs (displaced frame differences) - a process which is referred to as training the hyperplane system. Once the system is trained, the global motion between any two frames of a video sequence can be recovered from the DFD - as described in section 5.2.1. The training phase of the hyperplane system is performed on each frame whose motion is to be determined - as explained in section 5.2.2. Estimating motion with a hyperplane technique mainly involves the inverse operation of a square matrix of the size of the number of sampling points making the system computationally expensive. Section 5.2.3 shows that the hyperplane estimator can be used in two different modes of operations depending on the camera use and that in the best case, the estimator can operate in real-time but only for a limited range of image motions.

5.2.1 Hyperplane motion estimator: the estimation phase

Estimating motion with the hyperplane technique [76] involves first training the system by building a set of linear relationships between the DFDs and the corresponding motion perturbation parameters. After the linear system is trained, the estimation of the p unknown parameters of the motion vector $\mathbf{a}_t^\tau$ between frame t and frame τ is simply achieved from n grey-level differences

as follows

$$\mathbf{a}_t^T = M_h \Delta \mathbf{I}_t^T \tag{5.1}$$

where M_h, the *hyperplane matrix*, is the pxN matrix containing the parameters of the trained system, and $\Delta \mathbf{I}_t^T$ contains the n grey-level differences described by

$$\Delta \mathbf{I}_t^T = [\Delta I_t^T(\mathbf{x}_1), \ldots, \Delta I_t^T(\mathbf{x}_n)]^T \tag{5.2}$$

$$\Delta I_t^T(\mathbf{x}) = I_t(\mathbf{x}) - I_T(\mathbf{x}) \tag{5.3}$$

By definition, if there are p parameters to be estimated then there must be at least p different constraints. The DFD is sampled at n points whose size is in general much greater than p which is of maximum 8 for the planar motion model defined in section 3.2.2. Hence, having a hyperplane matrix, the motion between any two frames is expected to be correctly estimated by simply multiplying the hyperplane matrix with the corresponding DFD at the appropriate sampled locations. The building phase (also called the training phase) of the hyperplane matrix M_h is described in section 5.2.2.

5.2.2 The training (or perturbation) phase

Estimating motion with the hyperplane technique requires the system to be trained. This is performed by perturbing the system with a set of motions which are likely to cover the range of motions which might be encountered. The perturbation generates grey-level differences between a frame and the perturbed frame. Using equation 5.1 with N_h perturbing motion vectors $\mathbf{a}_i$ ($i = [1 : N_h]$) giving a set of N_h grey-level difference vectors $\Delta \mathbf{I}_i$:

$$\mathbf{a}_i = M_h \Delta \mathbf{I}_i \tag{5.4}$$

where each vector $\Delta \mathbf{I}$ contain the grey-level differences at the location of n samples in the frame:

$$\Delta \mathbf{I}_i = [\Delta I_i(\mathbf{x}_1), \ldots, \Delta I_i(\mathbf{x}_n)]^T \tag{5.5}$$

$$\Delta I_i(\mathbf{x}) = I_t(\mathbf{x}) - I_t(\mathbf{x} + X\mathbf{a}_i) \tag{5.6}$$

with X the matrix of the motion model as described in section 3.2. Using a least-squares regression scheme to minimise the N_h linear equations given by equation 5.4 gives the following functional

$$\epsilon(M_h) = \sum_{i}^{N_h} (\mathbf{a}_i - M_h \, \Delta \mathbf{I}_i)^2 = 0 \tag{5.7}$$

The hyperplane matrix is found when the derivative of the error ϵ with respect to M_h is zero and M_h is estimated as

$$M_h = \left(\sum_{n=1}^{N_h} \mathbf{a}_n \Delta \mathbf{I}_n^T \right) \left(\sum_{n=1}^{N_h} \Delta \mathbf{I}_n \Delta \mathbf{I}_n^T \right)^{-1} \tag{5.8}$$

Equation 5.8 requires the computation of the inverse of a $[n : n]$ matrix and two N_h summations of two matrices, since $\Delta \mathbf{I}$ is a large vector of n values (the sampling size). The computational

time necessary to compute the summations of matrices of equation 5.8 is expected to be relatively small compared to time required to compute the inverse. Inverse operations are computed here by the SVD (Singular Value Decomposition) technique [103]. This expensive computation is necessary each time the system needs to be trained *i.e.* for each new frame. The next section discusses the possibility of re-using the inverse matrix results between captured frames instead of re-computing this inverse operation whenever a new frame is captured. In summary, if the system requires a sampling size much greater than the number of motion parameters, the computational cost is shared by the sum of the two following processes:

1. Expensive inverse matrix operation, in equation 5.7, of a square matrix of size $[n : n]$ (where n is the sampling size)
2. Varying computational speed in terms of the number of perturbations N_h for the construction of the matrices implied by summations in equation 5.7.

5.2.3 Operational mode

The video sequences used as experimental datasets in this chapter are captured by surveillance cameras. When a camera is moving, the content of one frame captured at time t disappears progressively in time and the entire original content can disappear at worst within a few frames. The severity of the motion will dictate one of the operational modes. In the first *reference mode*, the original scene projected onto the reference frame disappears rapidly as the camera moves as illustrated in the left of figure 5.1. Here it is necessary to successively compute the motion between frames. In the second *dynamic mode*, the original scene is projected within each frame. In this case it is possible to compute the motion between each frame and the original reference frame as illustrated in the right of figure 5.1.

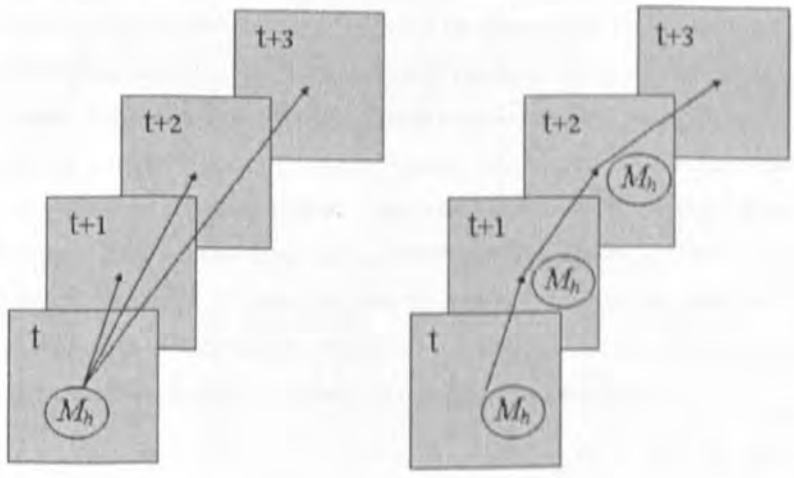

Figure 5.1: Reference (left) and dynamic (right) operational mode for hyperplane estimation

In figure 5.1, an arrow represents the global motion estimation phase (described by equation 5.1)

and the circled term M_h represents the expensive training of the hyperplane matrix (described by equation 5.8). Thus

- reference mode: $\mathbf{a}_t^\tau = M_h \Delta \mathbf{I}_t^\tau$ and the reference frame remains the same frame at time t for all frames and, hence, M_h is calculated only once,
- dynamic mode: $\mathbf{a}_\tau^{\tau+1} = M_h \Delta \mathbf{I}_\tau^{\tau+1}$ and the reference frame is different in each interval time τ, and, hence, M_h is calculated repeatedly.

In the dynamic mode, the hyperplane matrix must be re-trained for every new frame with a set of training motions of relatively small maximum amplitudes as will be shown in the experiments. In the reference mode, the hyperplane matrix is trained once with a set of motions whose extend must overlap the expected range of motions from the reference. A more intense training is hence required for the reference mode of operation although the cost involved in its matrix inversion is the same.

Hyperplane motion estimation is assumed to be performed at frame-rate whereas the hyperplane training is likely to be performed off-line because of the computational costs. In many situations, cameras are likely to be used within their full range of functionalities. For example, in surveillance applications, cameras often view a vast field of view in car-parks, offices, underground stations *etc.* Therefore the frame-rate estimation phase in the reference operational mode is unlikely to provide the required scene coverage without the repeated change of the reference image, and hence the requirement to rebuild the hyperplane matrix.

5.2.4 Summary

The hyperplane global motion estimation technique works in two phases: the training or perturbation phase (defined in section 5.2.2) followed by the estimation phase (defined in section 5.2.1). The hyperplane system, or more specifically the hyperplane matrix needs to be trained extensively before the system can estimate motion between pairs of frames. The training phase is computationally expensive due to a costly inverse matrix operation on a square matrix of the size of the number of sampling points. This training phase is an off-line (or non-real-time) operation whereas the estimation phase can be performed at frame-rate. The hyperplane's overall speed of execution within a video sequence is dependent on the operational mode of the camera (as described in section 5.2.3). However, but the hyperplane estimator is likely to be non-frame-rate when the camera is required to handle large motions.

5.3 Gradient-based global motion estimation

As a comparison, a gradient-based estimator is adapted to also estimate global motion parameters. A non-iterative version of the gradient-based estimator developed in chapter 3 is described

in this section. A non-iterative version is adopted for the comparison, as the hyperplane technique is also a non-iterative estimator. Recalling the optical flow expression developed in chapter 3 in equation 3.40 and 3.78:

$$\mathbf{a} = \mathbf{a}_0 + \sum_{k=0}^{K-1} \Delta\mathbf{a}_{k+1} \tag{5.9}$$

which consists of an initial motion $\mathbf{a}_0$ and K small motion $\Delta\mathbf{a}$ estimates each defined by

$$\Delta\mathbf{a}_{k+1} = \left(\sum_{\mathbf{x}'}\sum_{\tau} \varphi_\tau^T(\mathbf{x}', \mathbf{a}_k)\varphi_\tau(\mathbf{x}', \mathbf{a}_k)\right)^{-1} \sum_{\mathbf{x}'}\sum_{\tau} \varphi_\tau^T(\mathbf{x}', \mathbf{a}_k)\Delta I_t^\tau(\mathbf{x}', \mathbf{a}_k) \tag{5.10}$$

where

$$\varphi_\tau(\mathbf{x}, \mathbf{a}_k) \approx \frac{\partial I_\tau(\rho(\mathbf{x}, \mathbf{a}_k))}{\partial \mathbf{a}_k} \tag{5.11}$$

$$\Delta I_t^\tau(\mathbf{x}', \mathbf{a}_k) = I_t(\mathbf{x}) - I_\tau(\rho(\mathbf{x}, \mathbf{a}_k)) \tag{5.12}$$

$$\rho(\mathbf{x}, \mathbf{a}_k) = \mathbf{x} + \Delta\tau X(\mathbf{x})\mathbf{a}_k \tag{5.13}$$

There are n neighbouring pixels between m frames in the spatio-temporal window of estimation where $\mathbf{x}' = [\mathbf{x}_1 : \mathbf{x}_n']$ and $\tau = [\tau_1 : \tau_m]$. In this global motion case, there are no initial estimates available neither for the hyperplane nor the gradient-based technique. Hence $\mathbf{a}_0 = \mathbf{0}$ in equation 5.9. Moreover, for comparison purpose, the optical flow technique is chosen non-iterative *i.e.* $K = 1$. The estimation is performed only between two frames *i.e.* $\tau = t + 1$. Thus, the iterative motion expression from equation 5.9 becomes

$$\mathbf{a}_t^\tau = \Delta\mathbf{a}_1 \tag{5.14}$$

$$= M_j \Delta\mathbf{I}_t^\tau \tag{5.15}$$

where

$$M_j = \left(\sum_{\mathbf{x}'} \varphi_\tau^T(\mathbf{x}')\varphi_\tau(\mathbf{x}')\right)^{-1} \sum_{\mathbf{x}'} \varphi_\tau^T(\mathbf{x}') \tag{5.16}$$

$$\varphi_\tau(\mathbf{x}) \approx \frac{\partial I_\tau(\mathbf{x})}{\partial \mathbf{a}_k} \tag{5.17}$$

$$\Delta\mathbf{I}_t^\tau = [\Delta I_t^\tau(\mathbf{x}_1'), \ldots, \Delta I_t^\tau(\mathbf{x}_n')]^T \tag{5.18}$$

$$\Delta I_t^\tau(\mathbf{x}') = I_t(\mathbf{x}) - I_\tau(\mathbf{x}) \tag{5.19}$$

The computational cost of the gradient-based estimator in equation 5.14 is dominated by the construction of the matrix M_j (referred to in the chapter as the Jacobian matrix) as it contains mostly derivative terms. Similarly to the construction of the hyperplane matrix M_h, the cost of building the Jacobian matrix M_j is composed of the summations of matrices and the inverse matrix operation. However, the inverse operation is performed on a square matrix of size pxp where p is the number of motion parameters which is maximum 8 for the planar model. It is shown in section 3.7 that inverting such a small matrix is performed very quickly and that the computational cost of an optical flow estimation depends on the size of the neighbourhood: n. Therefore, the non-iterative gradient-based technique is likely to be performed at frame-rate

unless a very large number of sampling pixels is necessary.

Note also that the sampling pixels are usually drawn from a local neighbourhood around the pixel whose motion is being estimated. In the case of global motion, however, the sampling pixels could be drawn from the image in an arbitrary manner as discussed in the next section.

5.4 Evaluation procedures

In this section, the global motion existing between pairs of frames captured by surveillance cameras is estimated by two motion estimators: the hyperplane estimator introduced in section 5.2 and the non-iterative gradient-based estimator introduced in section 5.3. Section 5.4.1 presents the four surveillance datasets used for comparison. They are captured by cameras limited to pan and tilt operations only. In order to evaluate the performance of the estimators, the motion estimates need to be compared with the true motion parameters (called the ground-truth) which are typically unavailable. In section 5.4.4, an expensive correlation-based technique provides accurate translational vectors of the global motion which will substitute for our ground-truth. The hyperplane and gradient-based estimator are evaluated in section 5.5 against three main operational parameters:

- the smoothing factor σ (the width of the Gaussian kernel convolved with the input frames),
- the number of sampled points denoted n and,
- the sampling strategy that decides how to select n pixels of interest across the image frame. The different sampling strategies are described in section 5.4.2.

The performance of the hyperplane estimator depends not only on the three previous parameters but also on how it is trained. As described in section 5.4.3, the hyperplane matrix is trained using the two following parameters

- training range, denoted R_h, which describes the perturbation from each sampling point and is related to the maximum motion magnitude that may be estimated, and
- training gap, denoted G_h dictates the density of perturbations within R_h used to perform the training.

5.4.1 Data sets

Figures 5.2 and 5.3 present the four datasets (the 'lab', 'home-made', 'park-run' and 'car-park' sequences) captured by pan and tilt cameras, and used to evaluate the hyperplane and gradient-based estimators. Each pair of frames in each sequence undergo the global motion specified later in section 5.4.4. Table 5.1 presents the characteristics of the four sequences in terms of

sequence	'lab'	'home-made'	'park-run'	'car-park'
number of frames N_F	30	18	30	40
minimum motion magnitude	3	0	1	1
maximum motion magnitude	5	16	3.16	17
average motion magnitude	3.39	5.35	1.90	7.13

Table 5.1: input sequences characteristics

Figure 5.2: Example frame of the left: 'lab' and right: 'home-made' sequence

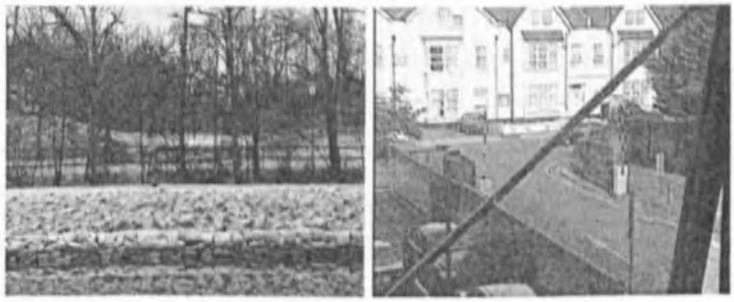

Figure 5.3: Example frame of the left: 'park-run' and right: 'car-park' sequence

number of pairs of frames N_F, minimum, maximum and mean motion magnitude among the N_F successive pairs of frames.

The 'home-made' sequence is a noiseless sequence created by cutting out sub-images from a same frame. Hence, the global motion is translational as all the pixels undergo the same motion. The 'lab' and 'park-run' sequences are captured by high-quality video cameras with low levels of noise. The 'car-park' sequence is captured by a poor quality surveillance camera creating motion blur on the image plane and accompanied by significant contrast variations. Moreover, this dataset is captured through a window whose external side is covered with a significant layer of dust generating high levels of different types of noise.

5.4.2 Locating pixels of interest: sampling procedures

Pixels should preferably be selected from edge structures where there exists enough local grey-level variability favourable for the gradient-based technique - chapter 3. The hyperplane technique can estimate motion given a sampled DFD (displaced frame difference) if the system is trained with such DFD. Therefore, the estimator is more likely to be performant when it is trained with unique DFDs likely to be present at edge structures.

Three sampling possibilities specify how n pixels are selected across the image. The first technique locates pixels on a grid, regardless of the grey-level content of the frame. The second technique, referred to as the edge-based technique, selects pixels randomly near gradient edges using a uniform spatial distribution [128]. The third region-based technique randomly selects pixels uniformly from non-edge pixels which contain low local gradient information.

The edge-based technique first smoothes the Sobel edge magnitude response of all the pixels. Second, weights are proportionally attributed between 0 and 1 where a weight of 0 corresponds to the lowest edge response and a weight of 1 is assigned to the strongest edge. Third, a pixel is selected randomly within the frame using a uniform technique and a second random number is generated uniformly between 0 and 1. If the weight associated with the edge response of the chosen pixel is greater than the second random number then this pixel is labelled as edge-based pixel. The same pixel cannot be generated more than once and this process is repeated until the required number of samples is obtained.

The three sampling strategies that will be compared are the grid technique (referred to as the *regular strategy*), the *hybrid strategy* which samples 50% of the pixels from edge pixels and the remaining 50% from the non-edge pixels, and 100% edge-based technique referred to as simply the *edge-based strategy*. It should be noted that the regular strategy samples pixels regardless of the content of the image, therefore small percentages of edge pixels are likely to be sampled by this technique. An example of 900 points are sampled by the regular strategy in figure 5.4. The same number of points sampled by the hybrid and the edge-based technique are shown on the left and right of figures 5.5 respectively.

A smoothing operation effectively flattens the structures within an image by smoothing local gradients and selected edge pixels may not lie very close to the actual gradient edges. Therefore, the distribution of sampled pixels recovered by the hybrid and edge-based strategy will vary with the degree of smoothing. On the other hand, the pixels sampled by the regular strategy remain sampled on the same grid regardless of the smoothing intensity. The examples in figure 5.5 are obtained with Gaussian smoothing operation with a kernel of width 0.5. An additional example is given in figure 5.6 for a smoothing width of 2.5.

5.4.3 Training the hyperplane matrix: the perturbation strategy

A reference frame needs to be perturbed by a set of motion parameters before any global motion estimation by the hyperplane technique is possible (as described in section 5.2.1). An optimal

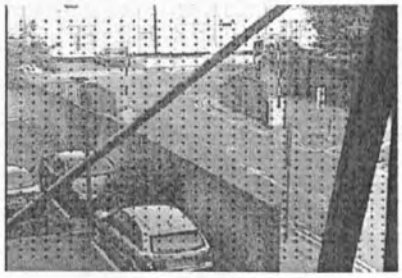

Figure 5.4: 900 regularly sampled pixels

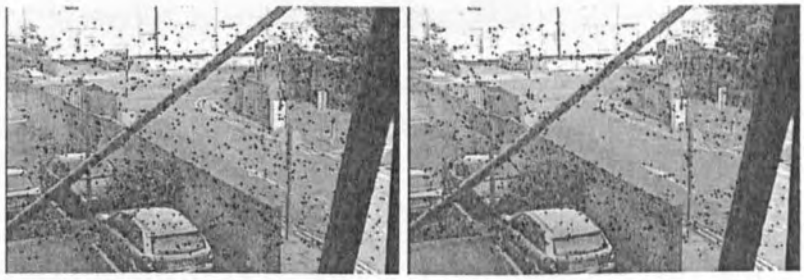

Figure 5.5: 900 points sampled by the left: 50% edge-based technique (hybrid strategy) and right: 100% edge-based technique (edge-based strategy) on a frame smoothed with $\sigma = 0.5$

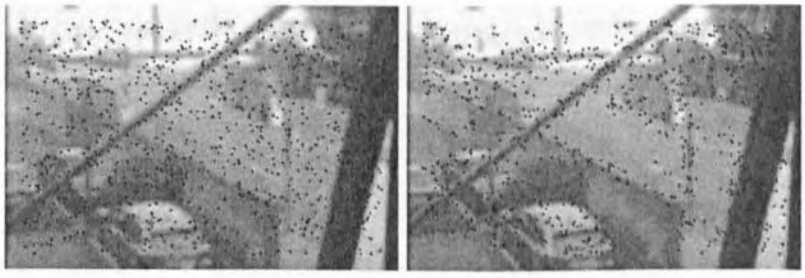

Figure 5.6: 900 points sampled by the left: 50% edge-based technique (hybrid strategy) and right: 100% edge-based technique (edge-based strategy) on a frame smoothed with $\sigma = 2.5$

training strategy would be to train the 8 parameters of the planar motion model (which best describes 3D scene motion) using a large sets of perturbed data. However, to limit the intensity of the training phase, the system is only trained with translational motions with different magnitudes around each sampled pixels.

The set of translational perturbing motion vectors are defined according to two training parameters: the training range, denoted R_h which is the maximum distance of perturbation and the training gap G_h which set the density with which the perturbations are performed. The number of perturbations N_h is given by R_h and G_h as

$$N_h(R_h, G_h) = \left(floor\left(\frac{2R_h}{G_h}\right) + 1 \right)^2 \tag{5.20}$$

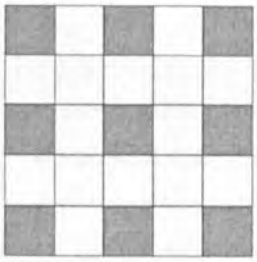

Figure 5.7: Window of 9 perturbations (shown in grey sampled pixels

where $floor(x)$ is the largest integer not greater than x. Figure 5.7 shows an example of $N_h = 9$ perturbations for a range $R_h = 2$ and for a gap of $G_h = 2$ pixels. The maximum motion magnitude present in the 4 experimental datasets is estimated to be 17 pixels. Thus a range R_h of 20 pixels is applied for the training of all the inputs. The training density D_h with which training is performed within R_h is given in percentages and is determined in equation 5.21 via the training gap, denoted G_h, that separates each perturbation: a gap of G means that the perturbation is done for translation increments of G pixels both horizontally and vertically *i.e.*

$$D_h(R_h, G_h) = \frac{N_h(R_h, G_h)}{N_h(R_h, 1)} = \frac{N_h(R_h, G_h)}{(2R_h + 1)^2}\% \tag{5.21}$$

Figure 5.7 shows an example of $N_h = 9$ perturbations with a density of D_h=28% (set by $G_h = 2$ and $R_h = 3$).

5.4.4 Generating ground-truth

The performance of the hyperplane and gradient-based estimators can be compared only if their results are evaluated against the true values of the motion parameters. Unfortunately there does not exist any way of providing the true motion parameters (also referred as to ground-truth) describing the global motion between any pair of images of the data sets. Therefore, in order to compare the two estimators, an expensive block-matching technique is developed which can

only provide the translational ground-truth components of the global motion between pairs of frames in the datasets.

A block-matching technique provides the ground-truth of the translating optical flow of the camera and works by finding the minimal cost function of a large template window (see section 2.2). The cost function implemented here is simply the mean squared of the grey-level difference between a large template window from one frame and the entire area of another frame as search area. The datasets do not contain any local motions caused by foreground scene objects except for the 'park-run' sequence which contain a low percentage of local motions. Thus no robust version is necessary and the least-squares block-matching is sufficient to provide accurately the ground-truth data for all the sequences.

5.4.5 Evaluation criteria

The evaluation of the performance of the gradient-based and hyperplane global motion estimator is performed using an error term measurement called the motion error similar to the one used in section 3.4.2. This motion error term is defined as the magnitude of the vector difference between the estimated and ground-truth global motion vector. The evaluation of both estimator is performed for two optimal parameters: the Gaussian smoothing factor σ, and the number of sampling points n and the motion error is, therefore, denoted $d_t(\sigma, n)$, and calculated by

$$d_t(\sigma, n) = \|\Delta \mathbf{x}_t(\sigma, n) - \Delta \mathbf{x}_t^g\| \tag{5.22}$$

where $\Delta \mathbf{x}_t^g$ is the ground-truth optical flow (between frame time t and the next captured frame $t+1$) and $\Delta \mathbf{x}_t(\sigma, n)$ is the corresponding estimated flow for a particular smoothing factor σ and sampling size n. The performance of one motion estimator (for fixed parameters σ and n) is measured by the mean motion errors from the N_F pairs of frames in a sequence as follows

$$M_d(\sigma, n) = \frac{1}{N_F} \sum_{t=1}^{N_F} d_t(\sigma, n) \tag{5.23}$$

The performance of the hyperplane estimator is evaluated against two additional parameters: the training range R_h and density D_h as described in section 5.4.3. Hence, the above evaluative terms are expressed with these two extra terms as follows: motion $\Delta \mathbf{x}_t(\sigma, n, R_h, D_h)$, motion error $d_t(\sigma, n, R_h, D_h)$ and mean motion error $M_d(\sigma, n, R_h, D_h)$.

5.5 Experimental results

The performance of the non-iterative gradient-based estimator and the hyperplane estimator described in the previous sections are first evaluated in section 5.5.1 below using the regular sampling strategy explained in section 5.4. These estimators are compared with two other sampling techniques, the hybrid and edge-based strategies, in section 5.5.3. The hyperplane estimator relies on two extra parameters compared to the gradient-based estimator which are

the training range and the training density. The optimal training strategy is evaluated in section 5.5.2. A discussion on the operational speed of the two estimators is found in section 5.5.4 and conclusions are drawn in the last section.

5.5.1 What is the best smoothing factor?

In the following two sections, experiments are run on the non-iterative gradient-based estimator and the hyperplane estimator using the regular sampling strategy. The data are plotted against the two main parameters: the smoothing factor σ and the number of sampling points n. The four datasets presented in the previous section 5.4 are used in this experimental section.

The gradient-based optical flow

In the following experiments the regular sampling procedure is used to locate training points across the image. The mean global motion errors M_d (see section 5.4.5) are plotted against various smoothing factors σ and various numbers of sampled points n in figure 5.8.

A minimum number of 20 samples is necessary before the results starts being stable in the example of the 'car-park' sequence shown in figure 5.8. Also, it can be seen that mean errors are at a minimum for smoothing factors σ in the range of [6-10] pixels (assuming $n > 20$ samples). The same behaviour for the motion errors is observed for the 3 other datasets with minimum errors found for smoothing factors in the ranges [1:5], [3:7] and [0.5-3] for the 'lab', 'home-made', and the 'park-run' sequences respectively.

A closer look of the optimal range of smoothing factors is found in figure 5.9 for the 'car-park' sequence. This figure displays the smoothing factor which gives the minimum motion errors for different sampling sizes. The graph shows that beyond 100 pixels a converged smoothing factor of 7 or 7.5 gives the best results irrespective of the sampling size. Similar results from the other sequences give optimal smoothing factors of [3-4], [4.5-5] and [1-3] pixels for the 'lab', 'home-made' and 'park-run' sequence respectively (again provided at least 100 samples are selected to ensure stability).

For the 'lab' sequence, the mean motion errors are plotted in figure 5.10 for optimal σ=7.5 against various sampling size. The graph drops rapidly to a minimum motion error of about 3.5 pixels. Repeating the same experiment on the other sequences produces the results displayed in table 5.2. The table displays for each sequence the mean motion magnitude, the optimal σ, and the mean motion error M_d. Note that the optimal σ is roughly equal to the motion magnitude. The mean errors are plotted against the mean motion magnitude in figure 5.11, and rises significantly as the motion grows.

The hyperplane optical flow

Similar experiments are performed on the hyperplane estimator *i.e.* against smoothing factor σ and the number of sampling points n (sampled by the regular technique). The hyperplane

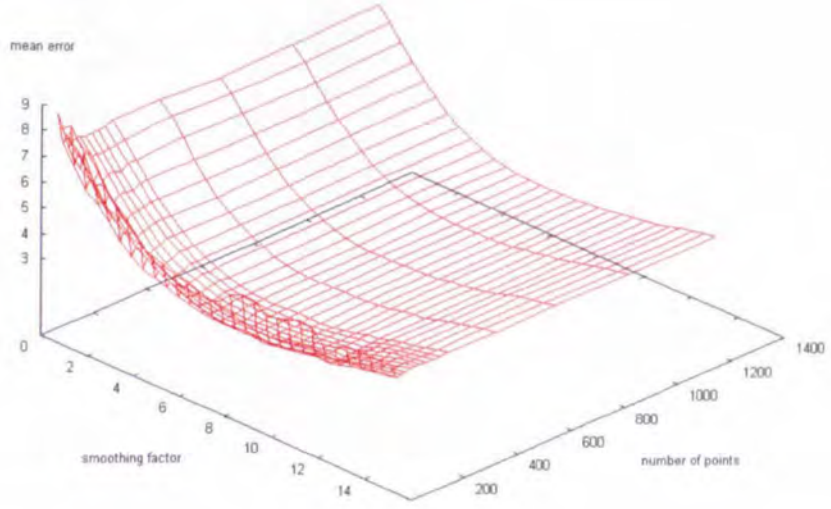

Figure 5.8: 3D plot of the global motion error of the non-iterative gradient-based estimator estimates for the 'car-park' sequence

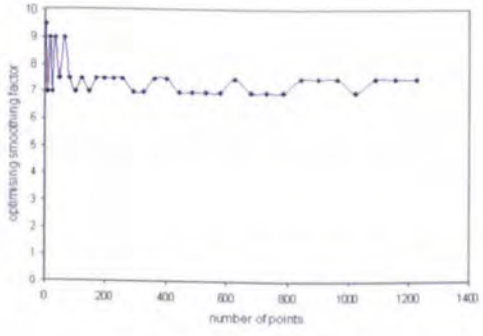

Figure 5.9: Optimising Guassian smoothing factor (σ=motion magnitude) of the gradient-based estimator for each sampled sizes for the 'lab' sequence

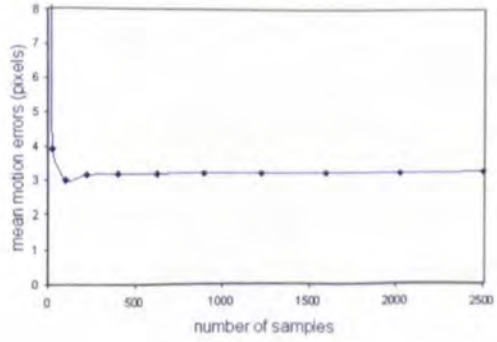

Figure 5.10: Optimal n of the gradient-based estimator with the optimal $\sigma = 7.5$ for the 'lab' sequence

sequence	'park-run'	'lab'	'home-made'	'car-park'
mean motion magnitude	1.90	3.38	5.35	7.13
optimal σ	2	3	5.5	7
mean motion error M_d	0.41	0.59	1.70	3.34

Table 5.2: Gradient-based results for an optimised smoothing factor σ and 100 sampling pixels

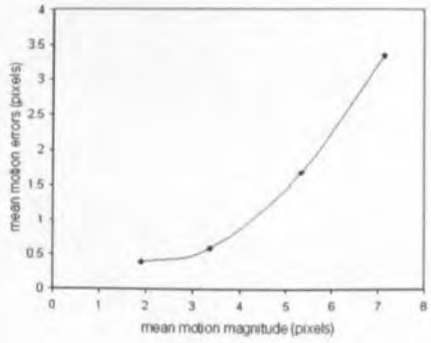

Figure 5.11: Mean motion errors of the gradient-based estimator against mean global motion magnitudes

technique is trained over a range of 20 pixels (the maximum motion magnitude present in the three sequences is 17 pixels) at a 100% density.

The hyperplane motion errors are plotted in figure 5.12 for the 'lab' sequence. The graph displays demonstrates that the performance of the hyperplane estimator becomes unstable and very inaccurate when the system is over-sampled and over-smoothed.

For different sampling sizes, the optimal smoothing factor σ is plotted in figure 5.13. The results show that the optimal σ converges to a value between 0 and 1 when there are more than 100 sampling pixels. The mean motion error is plotted against sampling size in figure 5.14 for three different smoothing factors *i.e.* σ =0, 0.5 and 1. These plots are all similar in that the mean motion error converges to less than 1 pixel for sampling sizes greater than 100 pixels. These experiments are repeated, as before, on the three other sequences and similar behaviours are observed. The best motion results are obtained in the following ranges of σ: [1-2], [0.5-2] and [0.5-2] for the 'home-made', 'park-run' and 'car-park' sequences respectively when their images are sampled with a minimum of about 81, 144 and 144 points respectively. As these ranges of smoothing parameter are similar in coverage, a common smoothing factor of σ =1 is chosen as the optimal σ for the hyperplane estimator. Moreover, the mean motion error can be assumed to have converged to a minimum for n=150 samples.

The mean motion error is displayed in table 5.3 for this sampling size of 150 pixels and a smoothing factor of 1 for all datasets. The mean motion error increases with the applied motion magnitude. This motion error is plotted in figure 5.15 for both the hyperplane and

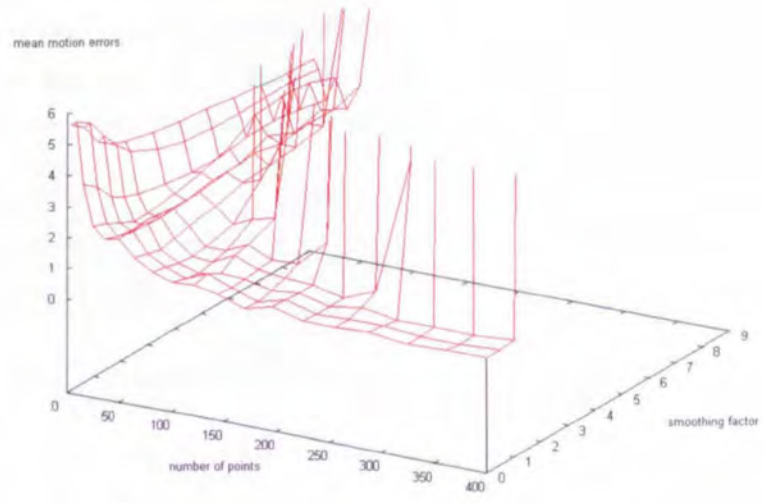

Figure 5.12: Hyperplane results for the 'lab' sequence at different scales

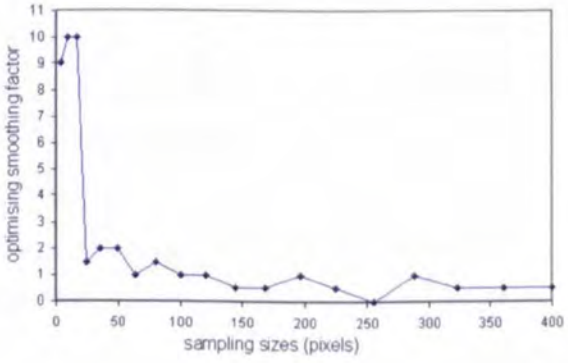

Figure 5.13: Optimal σ of the hyperplane estimator for the 'lab' sequence

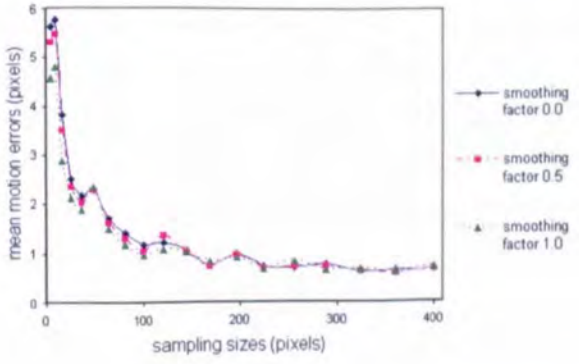

Figure 5.14: Optimal n of the hyperplane estimator for the 'lab' sequence

the gradient-based estimators as a function of motion. The graphs show that both estimators perform similarly when the magnitude of the motion is less than about 5.5 pixels. However, the gradient-based estimator's accuracy is better than the hyperplane's for larger motions. Beyond motion of four pixels in magnitude, both estimators estimate motion with an average error greater than 1 pixel from the ground-truth motion.

sequence	'park-run'	'lab'	'home-made'	'car-park'
mean motion magnitude	1.90	3.38	5.35	7.13
mean error M_d	0.51	0.85	1.68	4.90

Table 5.3: Hyperplane results for a smoothing factor of 1 and 150 samples

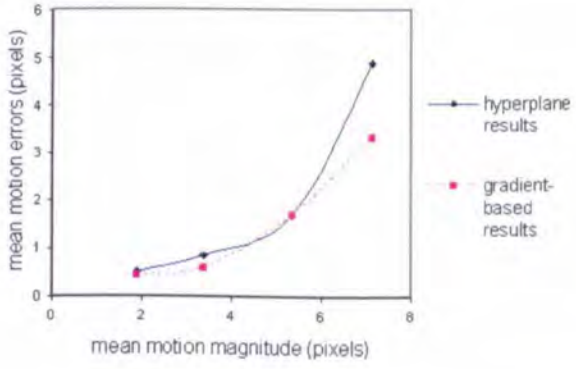

Figure 5.15: Mean motion errors against mean global motion magnitudes for the hyperplane and gradient-based estimators

5.5.2 What is the best hyperplane training strategy?

In the previous experiments, the training process of the hyperplane estimator is performed using the regular sampling strategy from a range of 20 pixels and with a training density of 100%. The results of section 5.5.1 show that a minimum number of 150 points is sufficient for an optimal smoothing factor of about 1 pixel. In this section, the hyperplane estimator's performance is evaluated for alternative training ranges R_h and densities D_h. First, the estimator is evaluated for different training ranges but with a 100% training density. Second, the training density is being varied for fixed training ranges.

Varying the training range

In this first experiment, the training range R_h is varied while the training density is fixed at 100% with the following optimal parameters: a smoothing factor of σ =1 and n =150 samples. For more rigorous analysis, the real motion existing between pairs of frames of a sequence is replaced by a set of 30 synthetically generated motions with varying magnitude *i.e.* 30 synthetically

generated frames are motion compensated from each frame of the sequence. The corresponding motion errors M_d (defined in section 5.4.5) are displayed in figure 5.16 for different training ranges R_h for the 'home-made' dataset.

Each motion error plot (corresponding to the following training ranges R_h =5, 10, 15 and 20 pixels) remains stable when the magnitude of the applied motion is lower than the respective R_h. Beyond this value, the motion error dramatically increases with the motion magnitude. Therefore, the hyperplane's stability is motion-dependent and it can only estimate motion magnitudes less than the training range R_h. It can also be observed that, over the stable range ($< R_h$), the hyperplane accuracy decreases as the training range R_h increases. The mean motion error over this stable range is tabulated against R_h in table 5.4. Therefore, not only the stability but also the estimator's accuracy is motion-dependent and the optimal training range is obtained if it set to the motion magnitude *i.e.* $R_h = \|\Delta \mathbf{x}^g\|$.

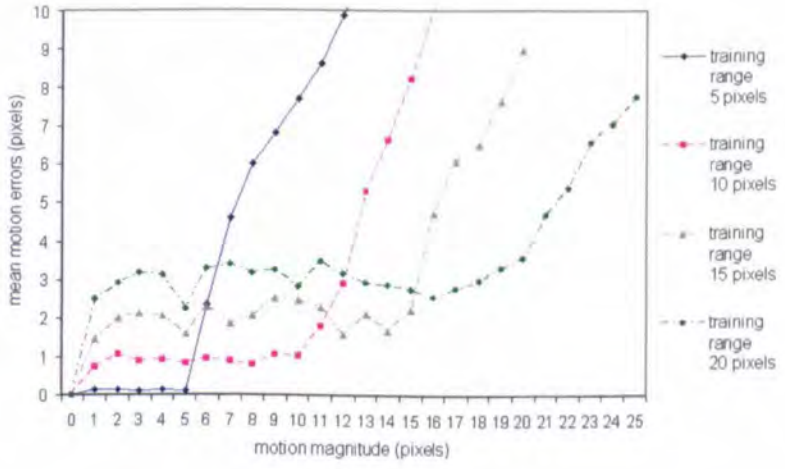

Figure 5.16: Mean motion errors for different training ranges against increasing motion magnitudes for the 'home-made' sequence

R_h	datasets			
(pixels)	'park-run'	'lab'	'home-made'	'car-park'
5	0.32	0.08	0.13	0.21
10	0.66	0.39	0.91	0.97
15	0.85	0.71	2.03	1.58
20	1.22	1.20	3.02	2.30

Table 5.4: Mean of the stable motion errors results

Varying the training density

The following experiment evaluates the hyperplane's performance when the density with which it is trained is altered. This training density was previously set to a maximum of D_h =100% and the motion errors M_d are here plotted for lower percentages. Figure 5.17 plots these motion errors for three different training ranges R_h for the 'lab' sequence example (with optimal smoothing factor $\sigma = 1$ and with $n = 150$ samples). The estimator's performance is maximum for training densities greater than 30% for the three different training ranges. For densities less than 10% the accuracy drops significantly.

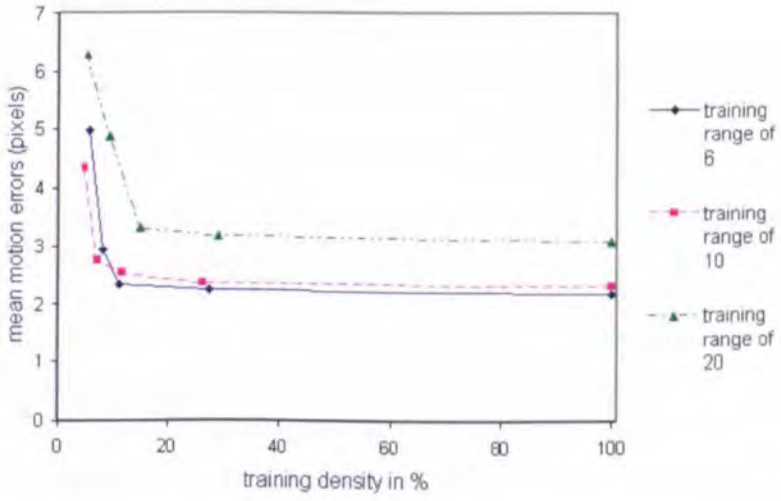

Figure 5.17: Performance of the hyperplane estimator for different training strategies with the 'lab' sequence

The first experiment, in section 5.5.1, showed that the hyperplane estimator remains stable and relatively accurate for a certain range of smoothing factors σ and sampling sizes n - see figure 5.12. The stability is plotted in figure 5.18 as a boundary between stability and instability regions for three training densities D_h for the 'lab' sequence example. The training range is set to 15 pixels (*i.e.* larger than the largest motion magnitude in the sequence). The stable region is located for the smallest pairs of σ and n and vice versa for the unstable region.

The area representing the range of optimal parameters σ and n where the system performs relatively accurately shrinks as the training density decreases. Note that for a small density of 13%, stability is met for sampling size just below the optimal size previously set to 150 pixels. Therefore, the hyperplane ought to be trained with a minimum percentage of 30% and a fully trained system with D_h=100% gives the optimal range of operations.

5.5.3 What is the best sampling strategy?

The previous experiments are performed with the regular sampling strategy described in section 5.4 which locates pixels on a grid across an image regardless of local content. The gradient-

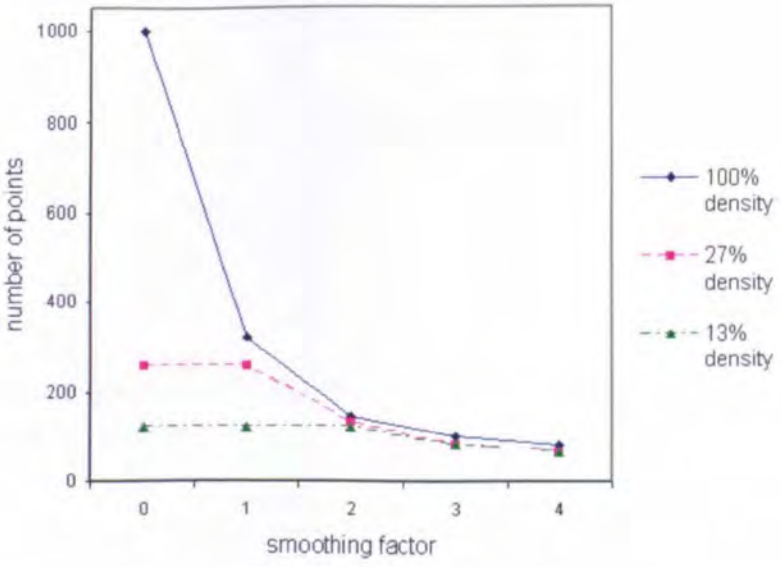

Figure 5.18: Hyperplane stability for various perturbation densities

based and hyperplane estimators are here evaluated using the hybrid and edge-based sampling strategies (also described in section 5.4). The hyperplane estimator is parameterised with the previously estimated optimal parameters: a sampling size of n =150 pixels, a smoothing factor of $\sigma = 1$ and a training range of R_h =20 pixels with 100% density. The gradient-based estimator is also parameterised with 150 samples but the optimal σ is set to the motion magnitude for each pair of frames.

The mean motion error M_d is displayed in table 5.5 for the gradient-based estimator and in table 5.6 for the hyperplane estimator when the regular, hybrid and edge-based sampling strategies are used.

Both techniques estimate relatively accurately the global motions of the 'lab' and 'park-run' sequences with a mean error below one pixel. These two datasets contain the smallest mean motion magnitudes with 3.38 pixels for the 'lab' sequence and 1.90 pixels for the 'park-run' sequence.

For bigger mean motion magnitudes (*i.e.* 5.35 pixels for the 'home-made' and 7.13 pixels for the 'car-park' sequence) both estimators give motion errors greater than 1 pixel. The error reaches a maximum value of 3.34 pixels for the gradient-based technique using the regular sampling strategy and a value of 5.18 pixels for the hyperplane estimator with the edge-based technique.

The average of the four mean errors of the four datasets for each sampling strategy in tables 5.5 and 5.6 are 1.50, 1.15 and 1.16 pixels for the gradient-based case and 1.99, 1.89 and 2.01 pixels for the hyperplane case when the regular, hybrid and edge-based sampling techniques are used respectively. The hybrid method gives slightly better results for both motion estimators which are plotted in figure 5.19. Both estimators perform relatively accurately with a mean

gradient-based mean motion errors			
dataset	sampling strategy		
	regular	hybrid	edge-based
'lab'	0.59	0.47	0.44
'park-run'	0.41	0.40	0.40
'home-made'	1.68	1.81	1.80
'car-park'	3.34	1.92	2.00

Table 5.5: Gradient-based results for different sampling strategies (with n=100 points and σ tuned to the mean velocity)

hyperplane mean motion errors			
dataset	sampling strategy		
	regular	hybrid	edge-based
'lab'	0.85	0.41	0.46
'park-run'	0.51	0.57	0.84
'home-made'	1.70	1.66	1.63
'car-park'	4.90	4.92	5.18

Table 5.6: Hyperplane results for different sampling strategies (with σ=1, D_h=100% and R_h tuned to the maximum velocity)

motion error of 1 pixel for sequences having a mean motion magnitude less than about 4 to 5 pixels. A motion error below half a pixel is obtained with both estimators if the motion to be estimated has a magnitude below 3 pixels.

5.5.4 Time analysis

It was previously shown that gradient-based and hyperplane techniques can estimate motion relatively accurately if motion magnitude does not exceed $\approx$3 pixels. For such small motion magnitudes, an optimised gradient-based estimator and a previously trained hyperplane technique can estimate motion at frame-rate. However, if the hyperplane technique requires to be trained for every captured frame, the training operation would render the estimation very computationally demanding. Training an hyperplane requires the inversion of a square matrix with lateral size n - the sampling size - which represents an $\mathcal{O}(n^3)$ operation. The gradient-based technique, on another hand, is a linear operation with the sampling size *i.e.* $\mathcal{O}(n)$.

The computational time required to estimate a 10 pixel motion by a hyperplane and a gradient-based estimator is plotted in figure 5.20 for varying sampling size. The rapid cubic time increase of the hyperplane training is observed in contrast to the slow linear growth of gradient-based estimator around one frame per second.

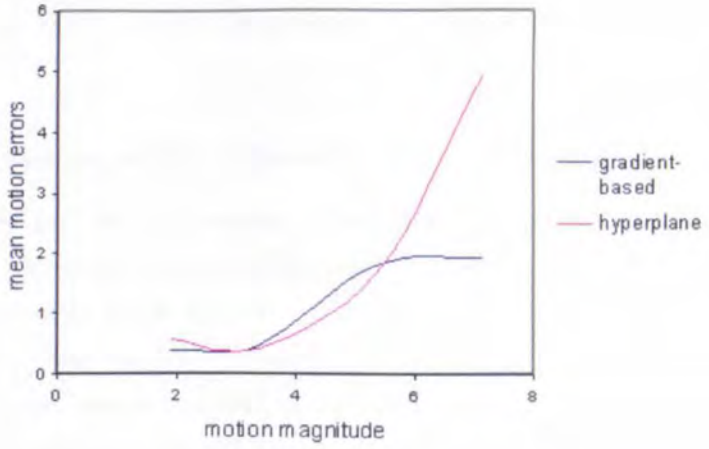

Figure 5.19: Gradient-based and hyperplane motion errors various mean motion magnitudes using the hybrid sampling method

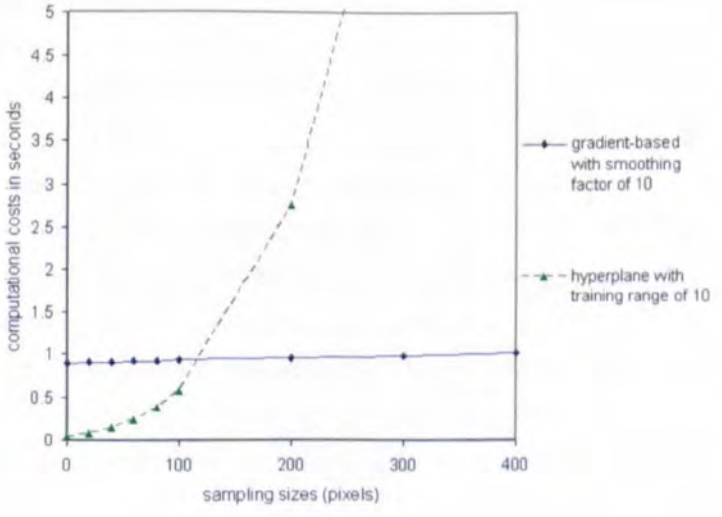

Figure 5.20: Computational costs of the gradient-based and hyperplane estimators when estimating a 10 pixel motion

For this large motion magnitude example, which is not accurately estimated by both estimators, the hyperplane's cost remains below the gradient-based's cost when there are less than approximately 120 samples. A optimal size of 150 samples was previously chosen. Therefore for large motion, the hyperplane estimation becomes very expensive and cannot be performed at frame-rate.

5.5.5 Discussion: which is the best motion estimator?

The non-iterative gradient-based estimator gives the best motion estimates if, before the estimation process starts, the grey-levels of the sampled pixels are pre-smoothed with a Gaussian kernel whose size is set to the unknown motion magnitude (as observed in chapter 3). The optimal number of sampling pixels is found to be approximately 150 pixels when sampled by the hybrid technique (described in section 5.4.2). The results showed that the gradient-based technique can only estimate, with relative accuracy, global motion with magnitude up to a few pixels - of about 3 pixels maximum with corresponding maximum mean motion error of half a pixel.

The hyperplane estimator performs similarly to the gradient-based estimator. However, it requires to be trained with a distance, or training range R_h, at least equal to the maximum expected motion magnitude in the dataset and with a training density of minimum 30%. The estimation was shown to become unstable for intense smoothing operations, *i.e.* large σ, and for large numbers of samples n. However, the errors on the motion estimates are minimum when 150 pixels are sampled from the image and when these pixels are weakly smoothed with a low smoothing factor of $\sigma = 1$.

Since both the gradient-based and hyperplane techniques estimate relatively accurately global motion with small magnitude, they can perform at frame-rate. If larger motions are to be estimated, and if training is necessary whenever a frame is captured (*i.e.* the dynamic mode - see section 5.2.3) then the inverse matrix operation required to train the hyperplane technique makes it very computationally expensive.

Therefore, the main limitation of the two global motion estimators evaluated in this chapter is the small motion magnitude limitation. Both estimators accuracy depends on this motion magnitude. Since straightforward techniques exist for the gradient-based estimator to address this problem, such as the multi-resolution approach developed in the previous chapter, the hyperplane is not adopted as global motion estimator in the remainder of this thesis.

5.6 Discussion

Estimating motion with the gradient-based estimator, developed in section 5.3, requires an inverse operation on a matrix with lateral size the number of parameters of the planar motion model *i.e.* 8 parameters. The alternative hyperplane method, developed in section 5.2, also

requires an inverse matrix operation during the training phase before motion estimation is achieved. However, unlike for the gradient-based case, this inverse operation is performed on a square matrix with varying lateral size given by the sampling size *i.e.* the number of pixels sampled across an image. If the global motion induced by the moving background scene relative to a pan and tilt camera is not too large, the motion estimation can be performed at frame rate using the reference operational mode described in section 5.2.3 where the system can estimate several motion from a unique trained system.

The results showed that the accuracy of both the gradient-based and hyperplane estimators are limited to small motion magnitude of a few pixels. When larger motions are to be estimated, accuracy decreases and is obtained when the pixels are smoothed with a kernel of the size of the motion magnitude for the gradient-based case and when the training amplitude is to be at least the size of this magnitude for the hyperplane case.

Since the hyperplane estimator can only recover small motion accurately, training with only small ranges are required meaning that this training can be performed at frame-rate and the dynamic model can also be performed at frame-rate. Therefore both estimators can use the dynamic mode with similar speed. The gradient-based estimator can address the small motion magnitude limitation using the multi-resolution approach developed in the previous chapter, as studied in the next chapter. Moreover, the gradient-based estimation can be implemented iteratively to improve its accuracy. The hyperplane, however, cannot be implemented easily in an iterative and multi-resolution framework and is not further investigated.

Chapter 6

Estimating motion of moving PTZ cameras

The two approaches previously implemented in chapter 5 - the hyperplane estimator and a gradient-based technique - recover motion with accuracies dependent on the unknown motion magnitude. For small motions, both estimators perform with relatively good accuracy and relatively high speed. Unlike the gradient-based estimator, the hyperplane cannot be easily implemented in an iterative and multi-resolution framework. The iterative and hierarchical gradient-based motion estimator developed in chapter 4 is adapted for global motion estimation in the first section 6.1. The gradient-based technique is successfully applied in this chapter to two applications. First, the motion of background scenes captured by moving pan and tilt cameras are estimated using the planar motion model described in section 6.2. While accuracy estimation is achieved to less than 1 pixel error, such accuracy is not sufficient for image registration applications such as background scene mosaicking. Second, in section 6.3, this estimator is then applied to a zooming application where the intersection between optical axis and image centre is estimated from the optical flow estimates.

6.1 Gradient-based global motion estimation

The dense robust and hierarchical estimator developed in chapter 4 is revisited in the first section. It is adapted for global motion estimation by using an edge-based sampling procedure described in the second section.

6.1.1 Hierarchical and robust approach

The small motion limitation of the gradient-based technique is addressed by the use of the hierarchical Laplacian pyramid technique described in chapter 4. The other major limitation of the gradient-based technique is caused by the presence of noise or motion outliers contaminating

neighbourhoods of pixels. Noisy outliers are usually introduced by the optical and electronic system of the camera and are easily eliminated by the robust statistical approach, also implemented in chapter 4. Motion outliers are mainly caused by occlusions which inject pixels undergoing different motions from the dominant motion of the majority of the pixels in a neighbourhood. Motion outliers cannot be totally eliminated by the robust technique.

In the two applications of this chapter, as the entire viewable 3D scene is assumed to undergo the same global motion, all pixels within the frame can be used in the estimation process. However, it was shown in chapter 3 that accurate optical flow only requires a minimum of pixels of the order of 100, and using all pixels in a frame would render the estimation computationally expensive. Moreover, optical flow estimation is dependent on the amount of grey-level structure present in the estimation. An edge-based sampling technique is therefore explored in the next section to select a desired percentage of the most contrasted pixels.

6.1.2 Edge-based sampling of the global neighbouring pixels

A sampling technique, referred to as the edge-based sampling technique, is here built to select the required number of pixels among the edges of a background scene for optimal optical flow estimation. This edge-based sampling technique is an alternative and slightly quicker version of the edge-based technique described in section 5.4.2 in the previous chapter which uses a random sampling approach. In the new version, edge pixels are found by local and oriented search of maximum edge response given by a Sobel gradient kernel. The image is divided by a grid and the search for a maximum gradient is performed along the thresholded edge pixels within the squares of the grid.

In most scenarios, datasets contain enough edges to distribute the minimum required number of pixels among them. For example, the frame on the left of figure 6.1 contains 28,500 edges represented as dark grey-levels on the right of the figure. The experiments in this chapter do not use more than 15,000 pixels. Figure 6.2 show an example of 1000 and 5000 sampled edge pixels of the input frame in figure 6.1.

6.2 Application to pan and tilt cameras

The motion estimates given by the global motion estimator described in section 6.1 are evaluated against two different sets of ground-truth vectors. The first set comprises of a large number of synthetically created translational vectors from which sequences are constructed. The second set comprises of the real motion existing between pairs of frames. The motion estimator is then evaluated against these two sets of motions in the two following sections.

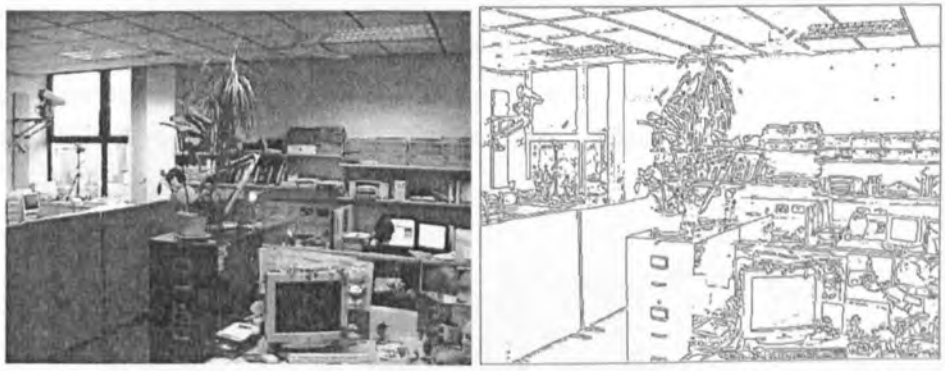

Figure 6.1: Left: input frame and right: its edges

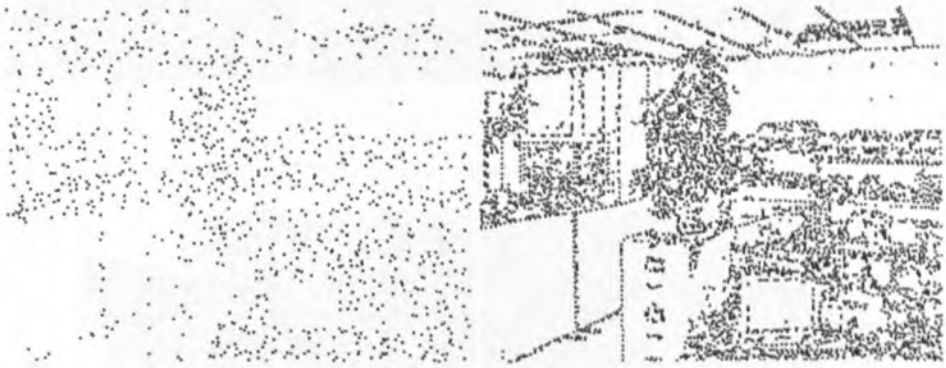

Figure 6.2: Left: 1000 and right: 5000 sampled edges among the edges of figure 6.1

6.2.1 Experiments with synthetical motions

An intense evaluation of the accuracy of the motion estimator developed in section 6.1 is performed in this section, using five input datasets. A motion error metric is defined for the evaluation of the performance of the estimator against varying motions.

Datasets and ground-truth

Five datasets are used to evaluate the gradient-based global motion estimator. Example frames are displayed in figure 6.3 for the 'lab', 'park-run' and 'car-park' sequence and in figure 6.4 for the 'lab2' and 'lab3' sequences, which contain 368, 251, 40, 200 and 200 frames respectively. All frames are composed of black and white 256 grey-levels. The 'lab' frames are constituted of 512x384 frames, the 'park-run' and 'car-park' frames are constituted of 384x288 pixels and the 'lab2' and 'lab3' frames are constituted of 384x256 pixels. The 5 sequences except the 'park-run' sequences are potential scenes for surveillance applications: 3 indoor scenes taken in different laboratories and 1 car park scene.

Figure 6.3: Frame example of the left: 'lab', middle: 'park-run' and right: 'car-park' sequence

Figure 6.4: Frame example of the left: 'lab2' and right: 'lab3' sequence

In order to test the maximum motion magnitude that can be accurately estimated, an intense evaluation is performed by generating sets of synthetically generated frames using known translational motions. For each frame in each dataset, a series of motion compensated frames is generated for 25 known translational vectors of increasing magnitude. These twenty five known

translational vectors are drawn from the magnitude range [1 : 25]. The direction of this motion is unique to each of the five datasets. In total, 26,475 motion estimations are performed. Note that these synthetical motions are translational only.

Metrics

The translational component $\Delta\mathbf{x}_t$ of the estimated planar model $\mathbf{a}_t$ (see section 3.2) is compared to the synthetical translational global motion $\Delta\mathbf{x}^g$ created for N_s pairs of frames via the following metric $d_1(\Delta\mathbf{x}^g)$ computed for each dataset:

$$d_1(\Delta\mathbf{x}^g) = \frac{1}{N_S}\sum_{t=1}^{N_S}\|\Delta\mathbf{x}_t - \Delta\mathbf{x}^g\| \tag{6.1}$$

where 25 sequences are generated for each of the 25 different motion magnitudes applied to all frames in a dataset.

It was demonstrated through experiment in section 4.2.5 that the motion estimator recovers accurately a wide range of motion magnitudes below a magnitude cut-off related to the number of layers in the hierarchical pyramid. At the top of a pyramid containing L levels, the maximum motion magnitude that can be accurately estimated is $V_{max} \approx 2$ to 3 pixels. Therefore, the maximum motion which can be handled at the base (original image) is $V_{max} \times 2^L$ *i.e.* for a 3 levels pyramid, motion of $\approx$ 8 to 12 pixels can be reliably recovered. This cut-off threshold will be evaluated and will allow the computation of the metric D_1, the mean of mean motion errors that the estimator can recover accurately, for each dataset:

$$D_1 = \frac{1}{V}\sum_{\|\Delta\mathbf{x}^g\|\leq V} d_1(\Delta\mathbf{x}^g) \tag{6.2}$$

Evaluation

The mean motion errors $d_1(\Delta\mathbf{x}^g)$ between the estimated global motion vector and the various ground-truth motions $\Delta\mathbf{x}^g$ are plotted in figure 6.5 for different ground-truth motion magnitude for each of the five datasets.

Similarly to the experiments performed in chapter 3, the global estimator was evaluated for different sampling sizes. The results showed that a minimum number of 100 pixels approximately is necessary before the motion estimates starts converging toward a final estimate as the number of samples increases. Between 500 and 1,000 pixels were necessary before the estimated motion magnitude is less than 10% from the converged motion magnitude, estimated to occur from 5000 pixels. A choice of 2,500 pixels is arbitrarily chosen as sampling size in the following experiments. All the motion estimates in figure 6.5 are very accurate provided the magnitude is below the maximum threshold cut-off magnitude.

However, the range of motion magnitudes over which reliable motion estimates can be estimated is typically larger than $V_{max}x2^L$. These individual ranges can be estimated by visually

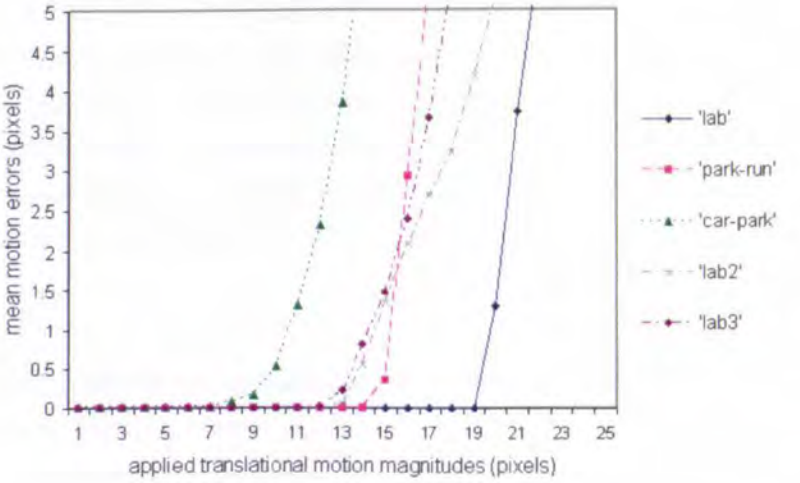

Figure 6.5: Global motion error results for six datasets

inspecting the plots of figure 6.5 to identify the *knee* point where accuracy dramatically worsens. These are tabulated in table 6.1 for each dataset where it can be seen that large motion magnitudes can be estimated with very small mean motion errors D_1.

Dataset	cut-off	D_1
'lab'	19	0.004
'park-run'	14	0.027
'car-park'	8	0.033
'lab2'	13	0.010
'lab3'	12	0.022

Table 6.1: Mean of errors of the accurately estimated global motion

6.2.2 Experiments with real captured motions

The motion estimator, previously evaluated against large sets of synthetical translational motion, is evaluated in this section against real captured motions. The same datasets are used for this evaluation. The ground-truth of the global motion is not available and an expensive correlation technique is instead developed to estimate the translational component of the real motion as described below. The motion estimates are finally evaluated against these real motions using a motion error metric.

Datasets and ground-truth

The same five datasets previously used to evaluate the motion estimator in section 6.2.1 are used in the evaluation against the real motion occurring between frames. Real global motions induced

by camera motion are modelled by a planar model. As there does not exist any straightforward technique to provide the ground-truth global motion, an expensive block-matching technique (the same used in section 5.4.4) is deployed to provide the translational global motion between each pairs of the sequences. The block-matching technique can only provide quantised estimation of the real translational motion. However this quantisation error can be neglected when averaging motion errors over a set of frames.

Metric

Global motion between frames captured between time t and $t+1$ are modelled by a planar motion model $\mathbf{a}_t$. From the estimated planar models $\mathbf{a}_t$, the corresponding translational global motion $\Delta\mathbf{x}_t$ (as explained in detail in section 3.2) is compared against the ground-truth motions.

A frame sequence is denoted S and contains N_S pair of frames and the evaluation is performed according to the motion error metrics explained below. The translational component of the estimated planar motion model is compared to the ground-truth translational component $\Delta\mathbf{x}^g$ of the real motion via the motion error metric d_2 defined by equation 6.3 for a sequence containing N_S pairs of frames.

$$d_2 = \frac{1}{N_S}\sum_{t=1}^{N_S} \|\Delta\mathbf{x}_t - \Delta\mathbf{x}_t^g\| \tag{6.3}$$

Evaluation

The mean motion errors d_2 defined in the previous section calculates the mean errors between the estimated global motion vector modelled from the real captured global motion and the ground-truth data are displayed in table 6.2 for the 5 datasets used previously. The results are also shown for a minimum of 2500 pixel samples recovered from the edge structures within each frame.

Dataset	d_2
'lab'	0.29
'park-run'	0.40
'car-park'	0.35
'lab2'	0.28
'lab3'	0.29

Table 6.2: Accuracy of the gradient-based global motion estimator

The maximum motion magnitudes present in each sequence are 5, 3.16, 9, 2 and 3 pixels for the 'lab', 'park-run', 'car-park', 'lab2' and 'lab3' sequences respectively. Each is less than the knee point tabulated in table 6.1. In table 6.2, the minimum deviation is between 0.28 and 0.29 pixel for the sequences taken in the same laboratory from different observation location *i.e.* 'lab', 'lab2' and 'lab3' sequences. The maximum deviations are given by the 'park-run' sequence

with 0.40 pixel in average from the ground-truth. In general, the deviations are always less than half a pixel from the ground-truth for the different datasets.

6.2.3 Conclusion

The gradient-based estimator evaluated through the previous experiments is shown to be an accurate global motion estimator for a large range of motion magnitudes up to between 10 and 20 pixels for different datasets. The local optical flow estimator developed in chapter 4 required a minimum number of 100 samples in its neighbourhoods of pixels whereas the global estimator of this chapter requires at least 2,500 pixels located on the edges for best performance. Despite this large number of samples, the estimator can still operate at frame-rate.

The global motion estimates are produced with an accurate mean error of less than 0.5 pixel. However these motion estimates are still not accurate enough for integrated image registration applications such as mosaicking. If image superposition is to be performed, this 0.5 pixel mean error would propagate and the mosaics fail to remain registered after a few frames.

6.3 Application to zooming cameras on static scenes

In this second global motion application, 3D static scenes are captured by non-moving cameras which are only changing their focal length *i.e.* zooming is performed on these scenes. This particular 3D motion relative to the camera is described in chapter 3 by the zoom motion model which allows the determination of the location of the *centre of expansion* (COE) which defines the intersection of the optical axis with the image plane. The gradient-based estimator described in section 6.1 estimates the parameters of the zoom motion model for three datasets presented with their respective ground-truth motion vectors in the second section. The results, displayed in the third section, show that the position of the estimated COE varies with the focal length *i.e.* the lens position is varying during the zooming operation.

6.3.1 Zoom motion model and the centre of expansion

Generally, cameras are manufactured such that the image centre roughly coincides with the middle point of the CCD array. The intersection of the optical axis with the image formation plane is commonly referred to as the image centre. This means that ideally, under zooming operations, the image centre should remain at the same location. However, changes of focal length results in changes in the camera parameters including the position of the image centre. The varying point of intersection between the optical axis and image plane is here referred to as the centre of expansion while the image centre remains fixed at all time at half the width and half the height of the image plane.

Although the scene in the field of view remains static, the change of focal length results in each pixel undergoing a relative motion depending on their location relative to the COE located

at $\mathbf{i}_o = [i_o, j_o]^T$ (in physical unit from the image centre) and the amount of variation in the focal length. The zoom model developed in chapter 3 gives the following relationship between the image velocity $\dot{\mathbf{x}}$ (also expressed in physical units) and its corresponding image location $\mathbf{x}$ measured from the COE

$$\dot{\mathbf{x}} = \frac{\dot{f}}{f}\mathbf{x} \tag{6.4}$$

where $\dot{f}$ is the amount of focal length f change per unit of time. The position $\mathbf{x}$ can be alternatively measured from the image centre $\mathbf{i}_o$ and becomes position $\mathbf{i} = [i, j]^T$: $\mathbf{x} = \mathbf{i}_o + \mathbf{i}$. This position can be converted into pixel unit as expressed in equation 6.5 where α and β are the constant factors introduced to convert distances from pixel unit to physical unit:

$$\mathbf{x} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} (\mathbf{i}_o + \mathbf{i}) \tag{6.5}$$

Replacing equation 6.5 in the image velocity equation 6.4 gives

$$\dot{x} = \frac{\dot{f}}{f}\alpha(i_o + i) = \frac{\dot{f}}{f}\alpha i_o + \frac{\dot{f}}{f}\alpha i \tag{6.6}$$

$$\dot{y} = \frac{\dot{f}}{f}\alpha(j_o + j) = \frac{\dot{f}}{f}\alpha j_o + \frac{\dot{f}}{f}\alpha j \tag{6.7}$$

or, in matrix notation

$$\dot{\mathbf{x}} = \begin{bmatrix} i & 1 & 0 & 0 \\ 0 & 0 & j & 1 \end{bmatrix} \mathbf{c} \tag{6.8}$$

where

$$\mathbf{c} = [c_0,\ c_1,\ c_2,\ c_3]^T \tag{6.9}$$

$$= \left[\frac{\dot{f}}{f}\alpha,\ \frac{\dot{f}}{f}\alpha i_o,\ \frac{\dot{f}}{f}\beta,\ \frac{\dot{f}}{f}\beta j_o\right]^T \tag{6.10}$$

The COE located at $\mathbf{i}_o$ can be estimated from equation 6.9 without the knowledge of the calibration factors α and β as follows

$$\mathbf{i}_o = \left[\frac{c_1}{c_0},\ \frac{c_3}{c_2}\right]^T \tag{6.11}$$

The global motion estimator developed in section 6.1 estimates the motion model vector $\mathbf{c}$ which allow the computation of the COE of equation 6.11 in the next two sections.

6.3.2 Experimental procedures

The global motion estimator described in section 6.1 is evaluated for static cameras performing zooming operations on 3D static scenes. Three zooming sequences are presented below with their ground-truth motions. The evaluation is performed by the use of a metric also developed below.

Datasets and ground-truth motion

The three datasets captured by zooming cameras are used for the evaluation where frame examples are displayed in figures 6.6 and 6.7. These sequences are the 'desk', 'vigilant' and 'paint' sequences, containing 90, 20 and 50 frames.

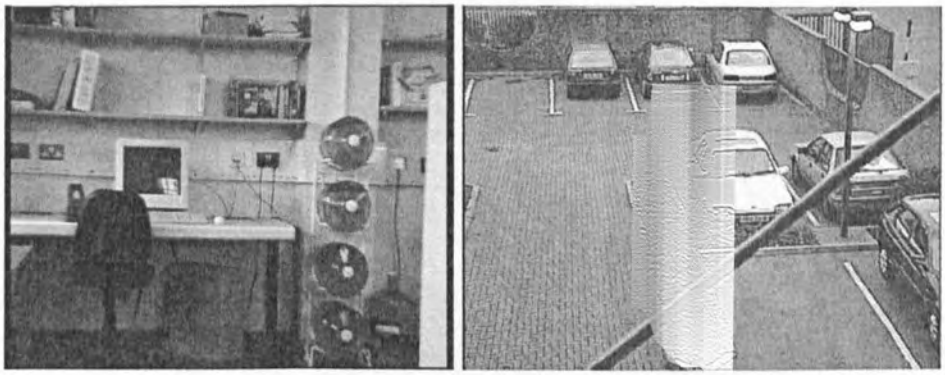

Figure 6.6: Frame example of the left: 'desk' and right: 'vigilant' sequence

Figure 6.7: Frame example of the 'paint' sequence

The ground-truth parameters $\mathbf{c}^g$ of the zoom motion model are generated by choosing randomly six parameters. These are the location of the centre of expansion $\mathbf{i}_o = [i_o, j_o]^T$ where image velocity is zero, and the maximum velocity $\dot{\mathbf{x}}_m = [\dot{x}_m, \dot{y}_m]^T$ that appear at a pixel close to the edge of the frame at $\mathbf{i}_m = [i_m, j_m]^T$. Using the motion model equation 6.8 at pixel locations $\mathbf{i}_o$ and $\mathbf{i}_m$ gives

$$\begin{cases} c_0^g i_0 + c_1^g = 0 \\ c_2^g j_0 + c_3^g = 0 \\ c_0^g i_m + c_1^g = \dot{x}_m \\ c_2^g j_m + c_3^g = \dot{y}_m \end{cases} \tag{6.12}$$

which gives as zoom model ground-truth solution

$$\mathbf{c}^g = [c_0^g, c_1^g, c_2^g, c_3^g]^T \tag{6.13}$$

$$= \left[\frac{\dot{x}_m}{i_m - i_o}, -\frac{i_o \dot{x}_m}{i_m - i_o}, \frac{\dot{y}_m}{j_m - j_o}, -\frac{j_o \dot{y}_m}{j_m - j_o}\right]^T \tag{6.14}$$

Metric

The metric $\Delta\mathbf{i}$ measures in equation 6.15 the distance between the estimated COE from its actual value in each frame of a sequence where g denotes the actual, or ground-truth data.

$$\Delta\mathbf{i} = \|\mathbf{i}_o^g - \mathbf{i}_o\| \tag{6.15}$$

6.3.3 Evaluation of the centre of expansion

The zoom motion model described in section 6.3.1 is applied to three datasets captured by zooming cameras. The motion model is estimated by the gradient-based global motion estimator developed in section 6.1 using the motion model defined in equation 6.9. The estimated centres of expansion are here evaluated using equations 6.11 and 6.15.

Figure 6.8 presents the cumulative percentage of estimated COEs as a function of error distance $\Delta\mathbf{i}$ for the 'desk', 'vigilant' and the 'paint' sequences. The graphs of figure 6.8 show that 100% of the COEs are accurately estimated with less than 1 pixel difference from their ground-truth locations for the 'paint' sequence. 100% of the estimated COEs of the two other datasets are accurately estimated if all the motion magnitudes are below approximately 10 pixels.

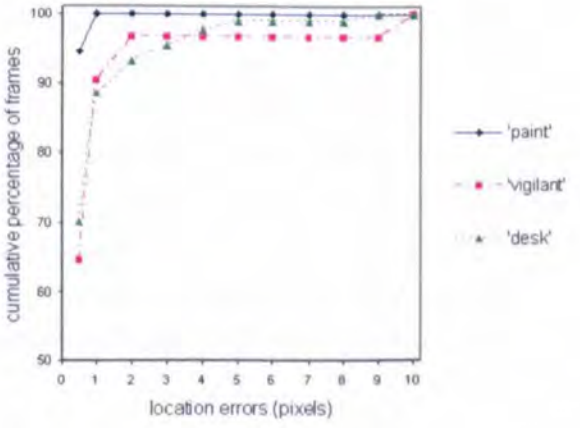

Figure 6.8: Cumulative percentages of COE location errors for various maximum allowed errors for 3 different sequences

The horizontal and vertical displacements of the COE from the image centre are plotted in figure 6.9. The 53 first frames are captured under a zooming operation performed on a desk alongside an office wall while the rest of the frames are captured with no focal length changes.

The COE is observed to move towards the image centre. However, this trajectory seems to oscillate in both the vertical and horizontal directions. This oscillation is thought to arise from mechanical asymmetries during the zoom operation. As soon as the zooming operation terminates, the estimation of the COE becomes unstable for frames 54-75. This is explained by the fact that if there is no focal changes, *i.e.* $\dot{f} = 0$ in equation 6.8 making the estimation impossible.

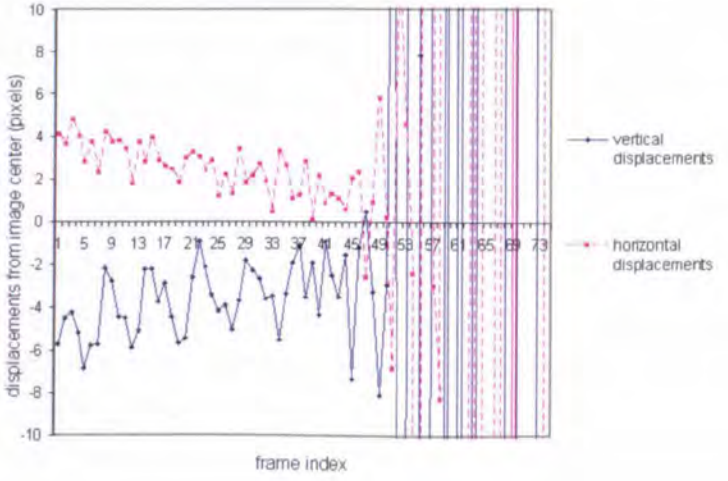

Figure 6.9: Vertical and horizontal displacements of the COE for the 'desk' sequence

6.3.4 Conclusion

The motion of 3D scenes captured by a camera performing zooming operations are modelled by the zoom motion model in section 6.3.1 which provides the formulation of the location of the centre of expansion (COE) *i.e.* the intersection of the optical axis with the image plane. The gradient-based global motion estimator recovers zoom models between pair of frames in three datasets in section 6.3.3 using a large number of 10,000 edge-based pixels. The results from synthetically created datasets show that the COE is estimated to within a few pixels deviation from its real position, and that in real scenarios, the COE's location does not remain fixed as the zooming operation changes *i.e.* the focal length changes.

6.4 Discussion

The global motion of 3D surfaces in distant scenes captured by PTZ cameras is estimated in this chapter by a gradient-based technique (section 6.1) and evaluated in a pan and tilt application in section 6.2 and in a zooming application in section 6.3. The gradient-based global motion estimator developed in the previous chapter adopts the hierarchical and robust iterative approach developed in chapter 4 to address the problem of noise and contamination by multiple motion.

In the first application, 3D scenes are captured by moving pan and tilt camera whose visual motion is modelled by a single planar motion model. Motion magnitudes with maximum magnitude between 10 and 20 pixels could be accurately estimated in datasets containing frames with about 300 to 400 pixels in width and height. Motion estimates are produced with an accuracy of less than 0.5 pixel deviation from the actual motions. However they are not accurate enough for integrated image registration such as background mosaicking applications. If image superposition is to be performed, this 0.5 pixel maximum error would propagate and background mosaics can not be constructed from the single estimated planar model.

Grabbing frames with a static camera performing zooming operations on a 3D static scene make the image velocity at every pixels undergo a single zoom motion model. Cameras are manufactured so that the intersection of the optical axis with the image plane, referred to as the centre of expansion, COE, is as close as possible to the image plane centre. It is demonstrated in section 6.3.1 that this COE is directly available from zoom model parameters. The gradient-based global motion estimator estimates these zoom models between pair of frames in three datasets in the second application using a large number of edge-based pixels, compared to the number of samples required in the pan and tilt application. The results with synthetically created datasets showed that the COE is estimated to within a few pixels deviation from its real position and that in real scenarios, the COE's location varies as the zooming operation changes the focal length, *i.e.* the mechanical operation performing zoom alters the position of the lens relative to the image plane.

Chapter 7

Tracking objects in surveillance applications

Visual tracking of objects has been extensively studied for many years. Nevertheless, the problems associated with tracking remain unsolved since there are many sources of ambiguities like shadows, illumination changes, over-segmentation of moving objects and object mis-detections. In addition, the high variability often present in the projected images of an object over time makes its tracking difficult. This variability arises from three principle sources: variation in the objects pose, variation in illumination, and partial or full occlusion of the target. When ignored, any of these sources of variability are enough to cause a tracking algorithm to lose its target or to mistrack with others. The traditional tracking approaches are reviewed in the first section 7.1. Tracking is mainly studied for surveillance applications using stationary cameras as studied in this chapter. The traditional approaches construct a background model of the captured frames of the static background scene over time. Foreground pixels are then extracted from the background model and groups of pixels are then formed. This object detection strategy is adopted in this study and described in section 7.2. The results show that tracking is successfully performed by simple data association rules but a different approach is necessary to track objects when their projected images are overlaping. This scenario caused by the occlusion problem is the focus of this chapter. Two trajectory model-based trackers and three appearance model-based trackers are designed in section 7.3 and compared in 7.6. Although the use of the appearance models improve the trajectory model-based tracking, additional information of the objects is needed for complete successful tracking during occlusion.

7.1 Review

The traditional way to track foreground objects is to segment moving objects which do not belong to a constructed model of the background scene as presented in the first section 7.1.1. The

background scene is built statistically over time and pixels which do not belong to the statistical models of the background are classified as foreground pixels which are then grouped spatially together to form the foreground objects also referred to as *blobs*. Establishing correspondence between the previously analysed objects with the newly detected blobs is the main difficulty in tracking efficiently these objects. In order to do so, *trajectory models* and *appearance models* of each objects are updated in time from the new observations. Traditional tracking is described in section 7.1.2 and section 7.1.3 focuses on the main contributions in the literature to improve tracking during occlusion.

7.1.1 Object detection

Traditional change detection algorithms segment foreground pixels from a constructed over time background of the stationary scene. Background models are usually constructed from either a mixture of Gaussian distributions of the intensities received at each pixel or from a Kalman filtering approach to model the background pixels. Other techniques are able to build models of the shadows caused by the foreground objects which are otherwise detected as foreground objects. The use of a Bayesian-based classifier (see Appendix C) can also allow for the segmentation of the foreground objects based on probability thresholding [60, 109].

Background modelling

Most of the tracking methods that perform change detection employ either a mixture of Gaussian models or a Kalman filtering method to model the background scene. Stauffer and Grimson [30, 122] have introduced the widely used concept of multi-Gaussian mixture model [46, 93, 106]. In this method, the grey-levels of the background reference image are modelled as a mixture of Gaussians and an on-line approximation is used to update the model whenever a new frame is captured. The Gaussian distributions of the adaptive mixture model are then evaluated to determine which models are the most likely to result from the background process. Each pixel is classified based on whether the Gaussian distribution which represents it most effectively is considered part of the background model. This mixture of Gaussians represents one the major contribution in adaptive background estimation. An improvement, presented in [78], has been added to the mixture of Gaussians technique. The method of Grimson *et al* [30] suffers from slow learning when the algorithm starts, especially in busy environments. In addition, it cannot distinguish between moving shadows and moving objects. The new approach improves the classical method by using a method based on EM (Expectation Maximisation) where the optimisation scheme used to fit a Gaussian mixture model is done by the EM algorithm.

The other major technique used to model the background image is based on a Kalman filtering approach [84, 85]. As performed with the mixture model method technique, the Kalman filtering technique also adapts to the changing illumination occurring in the background. Other approaches, such as [108], are acquiring statistics of image pixels (essentially mean and covari-

ance) in 3D colour space in order to build a confidence background map to guide the segmentation of the foreground objects. An interesting alternative way to estimate the background is presented in [109] where robust statistical filters model the background pixels using L-filters (i.e. a linear combination of the ordered grey-level samples of the image sequence).

Shadow suppression

Statistical-based methods are not only used to detect foreground pixels but they can also be used to overcome the problems caused by shadow [54, 78, 107, 109]. However, shadows often have the same characteristics as illumination changes of background surfaces and hence shadows can be modelled as semi-transparent regions for instance. To tolerate for the shadow, which are detected as foreground objects, frameworks raise a dedicated threshold. This threshold should be low if the variance of the estimated background values over time is low and vice versa.

Motion detection

The simplest way to separate the foreground pixels from the background pixels is to apply thresholds on the differences between the grey-levels of the background reference frame and the new ones of the captured frame *i.e.* the current frame. However, these empirical thresholds are often estimates of a probabilistic classifier which aims to calculate the highest probability that a pixel difference in absolute value belongs to a foreground, background or shadow pixel. The probability estimation is calculated using the Bayesian's rule *i.e.* a Bayesian classification. Using Gaussian distribution of the background noise, the threshold becomes easy to estimate by the use of a Bayesian classifier provided a reasonable estimation for the prior probabilities of background and foreground pixels are provided - Appendix C.

7.1.2 Object tracking

Typical tracking methodologies use a *hypothesise*, *validate* and *update* framework as illustrated in figure 7.1. The history of the information of each tracked object (position, velocity, blob dimension, colour-chromatic models *etc*) projects a prediction of this information onto the new incoming frame: the hypothesise phase. Each predicted object is compared with all new segmented foreground blobs from which a best match criterion validates the best candidate: the validate phase. Finally, the information of each object is updated with the new blob information: the update phase.

Trajectory model

Each active scene *object* has an associated *trajectory model* which describes the current position, velocity and possibly the acceleration of the object in image coordinates. An alternative coordinate system set on the ground plane may be a more appropriate space for tracking [106].

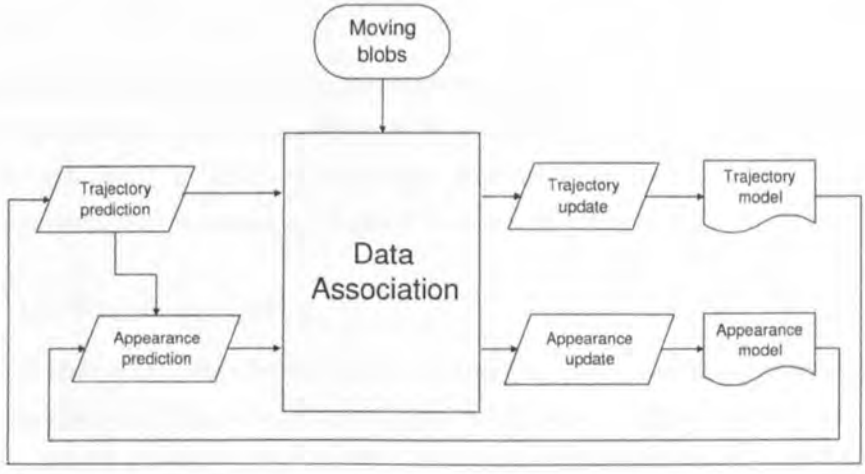

Figure 7.1: Tracking architecture

Modelling the trajectory of objects permits a better matching process between segmented objects and becomes of a great importance when several objects occlude each other and cannot be separated from the segmentation procedure. In many scenarios, the movement of objects can be predicted and a model can be approximated. For example, the direction of moving cars rarely change drastically from one frame to another and certain motion directions can be eliminated.

Appearance model

In addition, each *object* has an associated *appearance model* which may be used to identify these blobs with the most similar shape and/or chromatic structure. Such appearance models may simply describe the expected width and height of the object's bounding box, or record the pixel grey-levels within the bounding box. More sophisticated models may record the contour [116], binary pixel shape [73, 88], or spatio-chromatic structure [22, 31]. More *dynamic* variants of the appearance model may simply describe the rate of change of these bounding box dimensions [102, 105] while *active* or statistical appearance models may attempt to learn the variations in object appearances [72, 113].

Data association

Each active scene object, after being predicted onto the current frame, is *validated* by locating an appropriate corresponding observation - *i.e.* blob - from the list of candidate observations: this correspondence process is also referred to as *data association* [12]. *Greedy matching* is a common local approach for establishing correspondences where the closest observation to the predicted position of an object is chosen as best correspondence. In addition to incorporating appearance information, more sophisticated *global* approaches attempt to enforce the uniqueness constraint by considering all possible object-observation pairings [11, 101]. Unmatched observations may be used to hypothesise new objects appearing within the scene.

Updating

During the *update* phase, the position and appearance of each corresponding observation is used to update the trajectory and appearance model of validated scene objects. Typical update mechanisms include the $\alpha - \beta$ filter and the Kalman filter described in section 7.3. Fundamentally, the tracker maintains the temporal coherence of object identities.

7.1.3 Occlusion reasoning

The work carried out for visual surveillance contributed to improve the tracking correspondences when occlusions occur, also referred to as *occlusion reasoning*. Occlusion refers to two distinct processes: the *static occlusion* and the *dynamic occlusion*. Dynamic occlusion occurs when at least one foreground object occludes other foreground objects. Hence all these objects cannot be segmented into individual blobs by the traditional on-line hypothesise, validate and update method previously introduced. Partial occlusion is caused when a foreground object is being occluded by a static object belonging to the background scene. The generation of depth map of the background frame greatly improves tracking. For example, objects can be assumed to be moving on a ground plane only; enabling a depth ordering and a better estimation of the tracks [106]. However, the problem of correspondence is not as problematic in partial occlusion cases as in dynamic cases.

All existing trackers can only cope with moderate levels of occlusion, and most of them cannot cope well when moving objects leave the group of merging objects in different directions with which they entered. The longer an object merges into a group, the more difficult its tracking. While the use of appearance models, such as shape or chromatic texture models, are vital to establish temporal coherence of object identity, robust real-time implementations are not currently available for the computing platforms used in the visual surveillance research.

Alternative solutions to address the occlusion problem in tracking have been proposed. Rosales and Sclaroff [108] use a Kalman filter, Khan and Shah [81] segment object into similar colour classes and Colins *et al* [104] use a normalised colour histogram for each objects. Anzalone and Machi [5] use two combined methods. The first method uses a mixed parametric and fuzzy logic approach to compute distances among objects in feature space and to assign to each association an affinity index. The second method uses Kalman filtering. Recently, Haritoaglu *et al* [71] implemented a real-time human-tracking system and suggested a multi-camera system to analyse occlusions. Occlusion reasoning stages can consider the longer term history of each track to appropriately introduce merge and split operations and re-establish correspondence caused by occlusion or fragmentation [46].

7.2 Detecting moving objects

The detection of moving objects aims to generate bounding boxes, or blobs, of the moving foreground pixels. This detection is performed through two steps. First a statistical model of the background scene is constructed from all the captured frames via a multivariate Gaussian distributions technique and allow the segmentation of foreground pixels within a scene. This technique is referred to as the Stauffer and Grimson technique [30] and is decribed in the first section. The second step consists in grouping the foreground pixels into sets of individuals blobs. The technique descibed in section 7.2.2 projects iteratively the spatial distribution of the foreground pixels into histograms which are subsequently split into blobs. These foreground blobs are then associated to the corresponding objects in section 7.3.

7.2.1 Foreground segmentation

In most systems the first step in tracking objects is to separate the foreground from the background *i.e.* motion detection. This means to detect the regions of independently moving objects regardless of their speed, direction or texture. Moving objects are assumed to occlude a background captured from a single and stationary CCD camera with fixed focal length. This assumption is almost valid in most indoor scenes that are artificially lit. However, it is recommended to update the scene background actively to accommodate for variations caused by shadows or reflections that might yield false interpretation of events.

The background estimation process relies on identifying the parts of the image that belong to the stationary background in successive video frames. Hence, updating the reference image with the recent grey-level patterns is necessary to insure that it represents the manner in which the scene background is changing. Therefore, the estimation of the scene background relies on the robust classification of image parts as being foreground or background regions. An initial method for creating an adaptive background is to average the image grey-levels over time, creating a background approximation which is similar to the current static scene except where motion occurs. While this is effective where the background objects are visible for a significant portion of the time, it is not robust to scenes with high concentration of moving objects. It also recovers slowly when the background is uncovered by moving objects or when illumination in the scene changes. A thresholding technique can be then applied between the averaged background grey-level $Back_t(\mathbf{x})$ and the captured frame grey-level $I_t(\mathbf{x})$ to segment foreground pixels where $\mathbf{x}$ is a pixel location and t denotes the current time. For example, foreground pixels can be detected if the absolute difference between the frame and the background is greater than an empirical threshold: $|I_t(\mathbf{x}) - Back_t(\mathbf{x})| >$ threshold. This technique can be very sensitive to noise for a low threshold or can perform very poorly for a too high value of the threshold. A threshold on the contrast grey-level changes, *e.g.* $\frac{|I_t(\mathbf{x}) - Back_t(\mathbf{x})|}{Back_t(\mathbf{x})}$, is a more suited approach to overcome such problems, but it is nevertheless sensitive to light changes in dark regions of the image.

The technique used in this chapter to build the background reference frame uses the Stauffer and Grimson technique [30] to model the grey-level values of each pixel as a mixture of Gaussians. Based on the persistence and the variance of each of the Gaussians of the mixture, they determine which Gaussians may correspond to the background grey-levels. Pixel grey-levels that do not fit the background distributions are considered foreground until a Gaussian includes them with sufficient, consistent evidence supporting it. For example, if each pixel result from a particular surface under particular lighting, a single Gaussian would be sufficient to model the pixel values while accounting for acquisition noise. If only lighting changed over time, also a single, adaptive Gaussian per pixel would be sufficient. In practice, multiple surfaces often appear in the view of a particular pixel and the lighting conditions often change. Thus, multiple, adaptive Gaussians are necessary (typically around 5).

The multiple Gaussian system adapts to deal robustly with lighting changes, tracking through cluttered regions, and introducing or removing objects from the scene. Slowly moving objects take longer to be incorporated into the background. Repetitive variations are learned, and a model of the background distribution is generally maintained even if it is temporarily replaced by another distribution which leads to faster recovery when objects are removed. The implementation details are described in the remaining of this section.

The Stauffer and Grimson's reference background image is updated each time a new current frame is captured. Each background pixel is associated with a maximum of 5 Gaussian models. The Gaussian density grey-level function of mean intensity μ_t and standard deviation σ_t for the new captured grey-level $I_t(\mathbf{x})$ at location $\mathbf{x}$ is formulated as

$$\eta(I_t(\mathbf{x})|\mu_t) = \frac{1}{\sqrt{2\Pi}\sigma_t} e^{-\frac{(I_t(\mathbf{x})-\mu_t)^2}{2\sigma_t^2}} \tag{7.1}$$

Every new pixel grey-level $I_t(\mathbf{x})$ is checked against all the existing Gaussian distributions until a match is found. The matched model is the distribution which gives this minimum absolute difference $|I_t(\mathbf{x}) - \mu_{t-1}|$ provided that this difference is within three standard deviation σ_{t-1} of this distribution (where lies more than 99% of the data). If no matched distribution is found, the least probable distribution giving the least density function is replaced with a new distribution with mean value set to $I_t(\mathbf{x})$, an initially high variance is set to 3 and a low prior weight of 0.1. Each distribution is associated a weight which gives the proportion of the data that is accounted for each distribution. Hence, the more matched a distribution, the more important its weight. The weight of each distribution is updated at time t using the following equation

$$w_t = \begin{cases} (1-\alpha)w_{t-1} + \alpha & \text{if matched model} \\ (1-\alpha)w_{t-1} & \text{else} \end{cases} \tag{7.2}$$

where α is the learning rate of the algorithm of minimum value 0 and maximum value 1. Hence a matching model sees its weight increases quickly with high learning rate. The mean and

standard deviation of the matched distribution are then updated according to equations 7.3 and 7.4 respectively:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho I_t(\mathbf{x}) \tag{7.3}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(I_t(\mathbf{x}) - \mu_t)^2 \tag{7.4}$$

The learning rate is scaled by the following probability ρ of observing the new pixel in the distribution (equation 7.1)

$$\rho = \alpha\ \eta(I_t(\mathbf{x})|\mu_{t-1}, \sigma_{t-1}) \tag{7.5}$$

Unlike for the matched distribution, the remaining distributions are not updated and instead remain the same: $\mu_t = \mu_{t-1}$ and $\sigma_t = \sigma_{t-1}$ with a decreased weight (equation 7.2). Before the same process is repeated for the next captured frame grey-levels, the weights require to be normalised so that their sum equal 1 only if a new background model is added in the list of existing background models. The Gaussian distributions are then ordered by their decreasing value of w/σ. This ratio increases as a distribution gains more evidence (represented by w) or as its standard deviation σ decreases. The most 'dominant' distribution then locates first in the list of the existing Gaussian models and the least dominant locates last. This ordering process allows the selection of the B first distributions that represent the background scene models:

$$B = \text{argmin}_b \left(\sum_{k=1}^{b} w_t > T \right) \tag{7.6}$$

where T is a measure of the minimum portion of the data that should be accounted for the background and is here empirically set to 0.65. Finally, a pixel grey-level is assumed to be a foreground grey-level if it matches with any Gaussian distribution among the B first distributions. An example is shown in figure 7.2 where the left figure represents the 533^{th} frame in a sequence called the 'PETS1' sequence and the right figure displays the corresponding segmented foreground pixels in dark grey-levels.

Figure 7.2: Left: input frame and right: the foreground pixels (dark grey-levels)

In order to track foreground objects, their pixels require to be grouped into individual sets of pixels, referred to as *blobs*. The *connected-component* technique is a widely used technique

to extract these blobs, however, an alternative technique based on histogram projection [55] and described in the next section 7.2.2 is used in this application to extract these blobs.

7.2.2 Extracting moving regions: blob finding

After all the foreground pixels are segmented from the multi Gaussians background models previously described, the blob finding module developed in this section processes them into a set of *blobs*. The task of separating foreground pixels into different blobs is achieved by using a projected histogram method [55]. This technique creates histograms by projecting iteratively the spatial distribution of the foreground pixels on the horizontal and vertical axis of the blob bounding boxes. Having an original blob, an horizontal histogram of size the blob's width and a vertical histogram of size the blob's height are created. Each bin of the vertical histogram counts the number of foreground pixels across the width of the blob and the same operation is performed vertically for the horizontal histogram. One of these two histograms is then split into regions where a region is a set of consecutive bins having counts greater than 1. This blob is then split into several sub-blobs which are split according to the regions segmented from the second histogram. This two splitting procedures by the two histograms is iteratively repeated a few times until splitting is no longer possible. The examples in figures 7.3, 7.4 and 7.5 show that only 4 iterations is sufficient to segment the first blob chosen to represent the entire frame area. Blobs are segmented with a maximum of 10 iterations. Foreground pixel detection is often a noisy process resulting in small noisy blobs segmentation. In order to eliminate these noisy blobs, a threshold of 7 pixels minimum is performed on the width and height of each blob. Therefore if a blob lateral dimension is less than 7 pixels, it is eliminated from the list of valid foreground blobs. The results of this filtering process are shown in figure 7.5 where only 2 blobs remain as foreground blobs: one being the representative to the walking person and the other arises from foreground pixels segmentation noise.

Examples of segmented foreground blobs from figure 7.2 are displayed in figure 7.6. In this example the person walking from left to right has its head segmented into a noisy blob and hence eliminated from the list of valid objects, the van behind is segmented into blobs due to the static occlusion of the lamp post and the group of persons cannot be segmented from one merging blob. In order to avoid these erroneous blob splitting errors, the projected histograms are smoothed by a simple averaging process defined in equation 7.7 where $bin(i)$ is the number of foreground counts at the i^{th} bin of the histogram and the smoothed bin $bin'(i)$ becomes

$$bin'(i) = \frac{bin(i-1) + bin(i) + bin(i+1)}{3} \tag{7.7}$$

The histogram smoothing operation improves greatly the segmentation results as shown in figure 7.7 where the left walking person and the van are fully segmented into one blob. It can also be observed that the smoothing operation increases slightly the dimension of the foreground blobs. The group of persons cannot be split into individual blobs due to the occlusion between

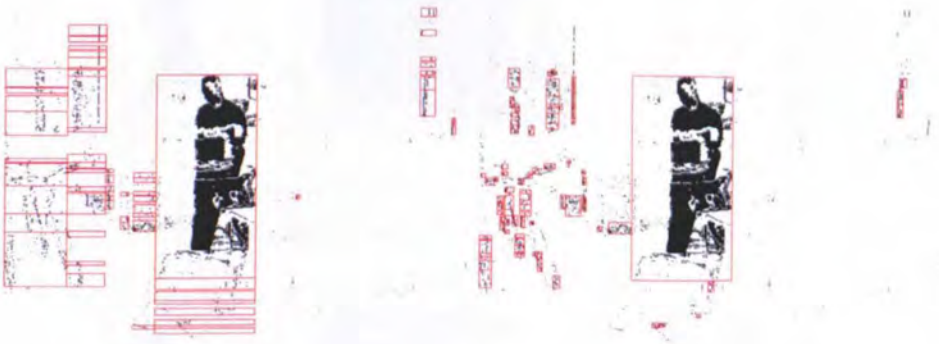

Figure 7.3: Blobs after left: 1 and right: 2 iterations of histogram projections

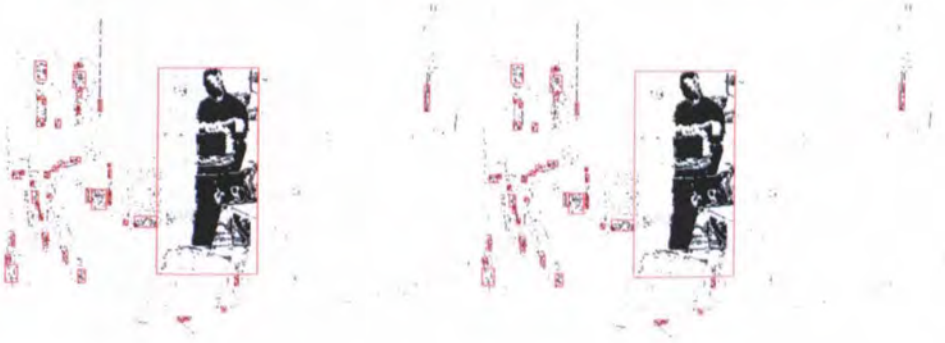

Figure 7.4: Blobs after left: 3 and right: 4 iterations of histogram projections

Figure 7.5: Foreground blobs after filtering

the left and the middle person and due to the shadows between the middle and right person segmented as foreground.

Figure 7.6: Segmented blobs bounding boxes of figure 7.2

Figure 7.7: Segmented blobs bounding boxes of figure 7.2 after histogram smoothing

7.3 Object tracking

The general aim of a tracking algorithm is to establish the temporal history of an object with reference to the set of feature observations (*i.e.* blobs) extracted from the image sequence over time, ideally from the moment each blob appears in the scene until it disappears. The performance of a tracking algorithm is directly dependent on the accuracy of the change detection between foreground and background objects. The multi Gaussian technique used to build the statistical background and to segment foreground pixels allows a reliable segmentation of the bounding boxes of the moving objects in a stationary scene. However, as explained earlier in this chapter, tracking objects in occlusion scenario remains a difficult task to achieve. Occlusion is detected when the predicted bounding boxes of at least two objects are overlapping *i.e.* their areas are intersecting, which positions are predicted from the previously captured frame.

Two widely used estimators, the $\alpha - \beta$ and Kalman filter techniques, predict object positions using previous observations and motion model regardless of the information present inside the objects such as grey-level pattern, depth or shape. They usually track object blindly during occlusion by the use of motion model assumptions. However, objects are easily mistracked as soon as their movements do not fit any predicted motion models.

The tracking algorithm, overviewed in the first section, predicts and updates object positions using two trajectory models described in the second section. Before trajectory models can be updated, correspondence between each tracked object and a new observed blob is established in the third section. Occluding objects are predicted in the last section.

7.3.1 Overview of the algorithm

The tracking algorithm is designed to be a multi-tracking algorithm for the purpose of evaluating and comparing during occlusion the two motion model-based filters (the $\alpha - \beta$ and Kalman filter) and three appearance model-based estimators (the correlation, region matching and hybrid estimator). Figure 7.8 illustrates the tracking algorithms where object positions are first predicted before being associated with a new observation during the data association process. In the case where a tracked object is predicted to enter an occlusion process where several tracks are associated with one same observation, three appearance model-based techniques attempts in the next section 7.4 to recover the lost information.

7.3.2 Trajectory modelling

As previously explained, tracking is performed via three steps: the prediction of the object position in the current frame, the correspondence between tracks and the new observed blobs and finally the update of the tracks trajectory models with the new observed positions. The motion model is chosen to have in this study a zero-acceleration and is implemented into an $\alpha - \beta$ and Kalman filtering process as follows.

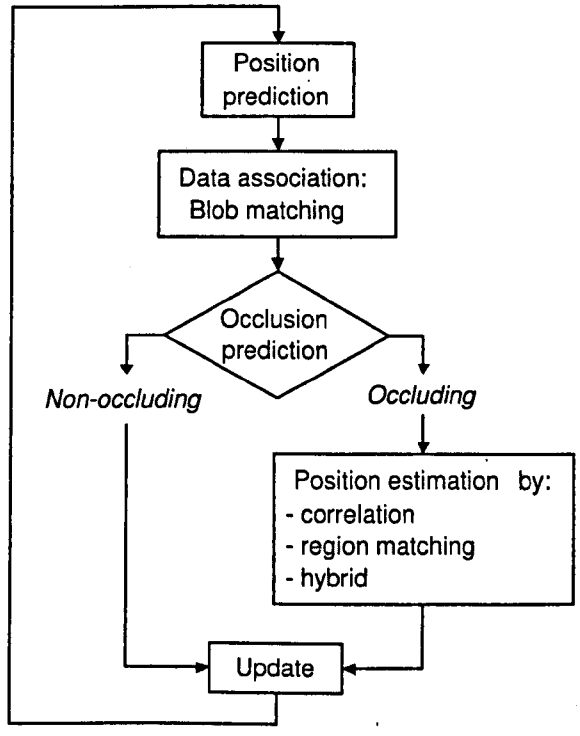

Figure 7.8: Evaluation procedure of the multi-estimators tracker

The $\alpha - \beta$ filter

The $\alpha - \beta$ and Kalman estimators predict the position of the tracked objects, each denoted B_t, at capture time t from their previous positions in the image (at time $t-1$) using the classical velocity $\mathbf{v}$ equation $\mathbf{v} = \partial\mathbf{x}/\partial t$ and the acceleration $\mathbf{a}$ equation $\mathbf{a} = \partial\mathbf{v}/\partial t$ respectively. A zero-acceleration model is used in this study meaning that velocities remain constant and objects positions are predicted with no acceleration influence as defined in the next equations

$$\tilde{\mathbf{a}}_t = \mathbf{0} \tag{7.8}$$

$$\tilde{\mathbf{v}}_t = \mathbf{v}_{t-1} \tag{7.9}$$

$$\tilde{\mathbf{x}}_t = \mathbf{x}_{t-1} + \Delta t\, \mathbf{v}_{t-1} \tag{7.10}$$

where Δt is the time interval between frames assumed constant and set to 1 and $\tilde{\mathbf{a}}$, $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{x}}$ are the predicted acceleration, velocity and position vectors respectively. The $\alpha - \beta$ filter, also known as $\alpha - \beta - \gamma$ filter updates the position, velocity and acceleration terms with a smoothing operation scaled by an α, β, and γ coefficient respectively. Because the acceleration is set to zero at all times, the updated position and velocity of the tracked objects are calculated using the new observed blob position $\mathbf{x}_t^*$ by

$$\mathbf{x}_t = \tilde{\mathbf{x}}_t + \alpha(\mathbf{x}_t^* - \tilde{\mathbf{x}}_t) \tag{7.11}$$

$$\mathbf{v}_t = \tilde{\mathbf{v}}_t + \beta\frac{(\mathbf{x}_t^* - \tilde{\mathbf{x}}_t)}{\Delta t} \tag{7.12}$$

In this study α is set to 1 and β is set to 0.7 meaning that the position is not smoothed with the predicted position but is always set to be the new observed position whereas the velocity is smoothed with a coefficient of 0.7.

The Kalman filter

The Kalman filter, described in detail in Appendix B, is a statistical estimator which estimates object positions according to covariance built over time from the errors recorded between the predicted and measured positions. Similarly to the $\alpha-\beta$ filter, the Kalman filter is implemented with the same zero-acceleration model. Recalling the formula of the predicted state $\tilde{\mathbf{p}}_t$ in appendix B and the predicted observation vector $\tilde{\mathbf{x}}_t$ respectively from the previous state vector $\mathbf{p}_{t-1}$:

$$\tilde{\mathbf{p}}_t = A\mathbf{p}_{t-1} \tag{7.13}$$

$$\tilde{\mathbf{x}}_t = H\tilde{\mathbf{p}}_t \tag{7.14}$$

where

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{and } H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{7.15}$$

with $\mathbf{p} = [x, u, y, v]^T$, the state vector, $\mathbf{x} = [x, y]^T$, the observation vector, and $\mathbf{v} = [u, v]^T$, the velocity vector. Given a known previous state $\mathbf{p}_{t-1}$ and a new observation of the object position $\mathbf{x}_t$, the state vector $\mathbf{p}$ and covariance P are updated by

$$\mathbf{p}_t = \tilde{\mathbf{p}}_t + K(\mathbf{x}_t - H\tilde{\mathbf{p}}_t) \tag{7.16}$$

$$P_t = \tilde{P}_t - KH\tilde{P}_t \tag{7.17}$$

where K is the Kalman gain defined by

$$K = \tilde{P}_t H^T (H\tilde{P}_t H^T + R)^{-1} \tag{7.18}$$

and the state covariance is predicted by

$$\tilde{P}_t = AP_{t-1}A^T + Q \tag{7.19}$$

and Q and R are respectively the covariance error of the state process and of the measurement process and are empirically set by the following equation where W and H are respectively the width and height of the rectangular blob.

$$Q \approx \begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix}, \text{and } R \approx \frac{1}{6}\begin{bmatrix} H & 0 \\ 0 & W \end{bmatrix} \tag{7.20}$$

7.3.3 Data association: blob matching

Blob matching is performed using a simple rule where a predicted track is associated with an observed blob which bounding box gives the largest area of overlap with the tracked object's bounding box. If there is no overlap with any of the observed blob, the tracked object is assumed to have entered a static occlusion (the object is totally occluded by objects belonging to the static background). In such cases, the tracks positions are updated by the trajectory model until the object reappear in the scene. In the case the predicted object does not match the re-appearing object, a new track is instead created and the old track is eliminated after a time-to-live threshold.

7.3.4 Occlusion prediction

Segmented blobs represent groups of foreground objects that cannot be separated into individual objects when occlusion occurs. Occlusion is here detected when at least two predicted tracks bounding box overlap with each other. A track is in an occlusion scenario when it is either occluding one or several predicted tracks or it is being occluded by one or several predicted tracks. Hence, the occlusion problem causes correspondence ambiguity in establishing correspondences between several tracks with the same observed blob. This matched blob that merges several objects is here referred to as the *merging blob*.

7.4 Recovering occluded observations using appearances

When a tracked object is not in an occlusion scenarios (no bounding box overlapping with other tracks - section 7.3.4), this track's position is updated with the new observation according to trajectory models described in the previous section. However, when occlusion occurs, three appearance model-based tracker attempt to recover the information lost in the merging blob. These three estimators predict positions by the use of the grey-level information present in the merging blob and are the

correlation technique, the

region matching technique and the

hybrid technique.

The positions of the tracked objects in occlusion are then updated with one of these appearance model-based estimators as described in the three following sections.

7.4.1 Correlation estimator

For each tracked object, $\widetilde{B}_t$ denotes its predicted bounding box located at the predicted position $\widetilde{\mathbf{x}}_t$ and B_t^* denotes the bounding box of the corresponding observed blob in current image

(matched by the data association process of section 7.3.3) and observed as a merging blob (from occlusion prediction in section 7.3.4). The correlation estimator predicts the actual track position $\mathbf{x}_t^*$ by the pixel position that gives the minimum least-squares match as follows

$$\mathbf{x}_t^* = \text{argmin}_{\mathbf{x}\in\mathcal{S}_t}\left\{\frac{1}{\text{Card}(\mathcal{W})}\sum_{\mathbf{x}'\in\mathcal{W}}\left(T_\tau(\mathbf{x}') - I_t(\mathbf{x}_t+\mathbf{x}')\right)^2\right\} \tag{7.21}$$

where $\mathcal{S}_t$ is the set of pixels within the search window, $\mathcal{W}$ is the set of pixels within the correlation window of size Card($\mathcal{W}$) pixels and T_τ is the template frame constructed from the grey-levels of the object bounding box before entering occlusion at time τ. The search window $\mathcal{S}_t$ is defined as the set of current foreground pixels $\mathcal{F}_t$ which lies within the intersection of the bounding boxes of the observed merging blob B_t^* and the predicted object bounding box $\widetilde{B}_t$ of the current object:

$$\mathcal{S}_t = \mathcal{F}_t \cap \left(\widetilde{B}_t \cap B_t^*\right) \tag{7.22}$$

where $\widetilde{B}_t$ and B_t^* represent the set of pixels within $\widetilde{B}_t$ and B_t^* respectively and $\mathcal{F}_t$ is then the set of foreground pixels within B_t^*. The set of pixels within the correlation window $\mathcal{W}$ is defined by all the foreground pixels within this window. If $\mathcal{M}_\tau$ is the set of foreground pixels in the frame captured before occlusion *i.e.* at time τ and if $\mathcal{T}_\tau$ is the set of pixels within the template T_τ then $\mathcal{W}$ can be expressed by

$$\mathcal{W} = \mathcal{M}_\tau \cap \mathcal{T}_\tau \tag{7.23}$$

7.4.2 Region matching estimator

The region matching technique predicts position during occlusion using the grey-level content of objects. The correlation technique (described in the previous section) is a least-squares pixel-based technique that can be easily influenced by noise. The region matching estimator is developed as a weighted region-based technique so it is less sensitive to noise. When an object is being occluded by another, its appearance, or here the grey-level content of objects, is also being from partially to totally occluded. The idea of the region matcher is to improve tracking by attempting to match the regions of the occluded objects surviving during occlusions. The region matcher consists in segmenting the various grey-level distributions of an object according to the range of textures held in the foreground pixels distribution as described in the first section. The second section describes how matching between regions of tracked objects and the regions of objects is achieved. The third section finally described how the tracked object position is predicted from a sum of weighted region velocities given the results of the previous matching process.

Region segmentation

A matched blob at time t is segmented into N' regions by a grass-fire technique, also referred to as a connected-component technique [112], and each region γ_k', where $k = [1 : N']$, consists of three features: the mean μ_k' and variance $\sigma_k'^2$ of the region's foreground grey-level and the

area A' of the foreground pixels distribution. The segmentation algorithm first computes for each foreground pixel the local grey-level variance in a 3x3 neighbourhood. The first region is created from the foreground pixel having the lowest variance. This region is then grown by the grass-fire technique with pixels having a grey-level belonging to the same Gaussian model respecting equation 7.24:

$$|I_t(\mathbf{x}) - \mu'_k| < 3\ \sigma'_k \tag{7.24}$$

Once a region is fully grown, the next region is grown from the non-visited pixel having the smallest variance. The process is repeated until all pixels have been grouped into regions where each region's features are updated with the grey-level and positions of the new introduced pixels. Regions having area with width and height less than seven pixels are eliminated from the list of regions as they are most likely to represent noisy or trivial regions. Figure 7.9 displays ellipses representing the second-order spatial moments of the regions segmented from the synthetic frame 'Mr Perfect' created by uniform grey-level distributions and from the real captured foreground person 'Mr Real'.

Figure 7.9: Regions are correctly segmented for synthetical regions of 'Mr Perfect' (left). Poor segmentation of 'Mr Real' (middle) is obtained (right).

In this figure, the pixels belonging to a region are displayed with an intensity equal to the mean value of the estimated Gaussian model. The segmentation results shows that, unlike for 'Mr Real', the body parts of 'Mr Perfect' are 'perfectly' segmented into their corresponding regions. Three consecutive frames of a foreground moving van and three foreground persons are shown in the figures of 7.10 and 7.12 and their corresponding regions are displayed in the figures 7.11 and 7.13 respectively. The results show the sensitivity of the segmentation process with real captured objects where small regions do not persist in time whereas large regions are likely to be segmented in several consecutive frames. Shadow cast by foreground object on the background scene is also a problem as it can represent a significant percentage of the total coverage of the regions. The shadow problem often fuses regions of objects as seen for example in the middle frame of the segmented van (figure 7.11).

Figure 7.10: Three consecutive frames of the moving van (see figure 7.2)

Figure 7.11: Segmented regions of the van

Figure 7.12: Three consecutive frame of the group of persons

Figure 7.13: Segmented regions of the three persons (figure 7.10)

Region matching

Correspondences between the regions of a tracked object and the regions of the matched observed blob are achieved by the region matching process. Object regions segmented at time τ (before occlusion occurs) are matched in each subsequent frame during occlusion at times indexed by t with the regions extracted from the observed merging blob. The k^{th} region of the tracked object is compared to the l^{th} region of the current blob by the mean of the similarity measurement S defined by

$$S_k(l) = s(\mu_l'; \mu_k) + s(\sigma_l'; \sigma_k) + s(A_l'; A_k) \tag{7.25}$$

where the similarity for each feature is calculated as follows

$$s(\alpha, \beta) = \frac{\min(\alpha, \beta)}{\max(\alpha, \beta)} \tag{7.26}$$

and where μ, σ^2 and A are the features of the tracked blob (described earlier) and μ', σ'^2 and A' are the features of the observed blob. The best match M_k of the k^{th} tracked region is the new region giving the following maximum similarity measurement:

$$M_k = \max_{l=[1:N']} \{S_k(l)\} \tag{7.27}$$

Object position estimation

The position of a tracked object in an occlusion scenario is estimated from the match results M_k (see equation 7.27) of all the regions γ_k with $k = [1 : N]$. The actual position is estimated as a sum of weighted displacements between occlusion time t and the time preceding the occlusion time τ:

$$\mathbf{x}_t^* = \mathbf{x}_\tau + \sum_{k=1}^{N} W_k \left(\mathbf{c}'_{m_k} - \mathbf{c}_k\right) \tag{7.28}$$

where $\mathbf{x}_\tau$ is the object position before occlusion, $\mathbf{c}_k$ is the position of the k^{th} region of that object, W_k is a matching weight defined in equation 7.29 and $\mathbf{c}'_{m_k}$ is the m_k^{th} observed position of the region which gives the best match M_k (equation 7.27). The weighting of the regions of the tracked objects is calculated in equation 7.29 and the weights are then normalised so that their sum over all regions is equal to 1.

$$W_k = \frac{M_k}{\sum_{i=1}^{N} M_i} \frac{A_k}{\sum_{i=1}^{N} A_i} \tag{7.29}$$

The weighting process in equation 7.29 is performed so that large regions (large A) having a high similitude with their matched blob's region (high M) participate highly in the estimation of the object position. Small size regions or regions with weak similitude with a region of the observed blob are poorly weighted and contribute less to the position estimation.

7.4.3 Hybrid estimator

The hybrid estimator is designed to improve tracking in occlusion scenarios by combining a trajectory model-based tracker with the region matching tracker (previously described). The

trajectory model-based estimator is either the $\alpha - \beta$ or Kalman technique. They both blindly tracks objects during occlusion according to a predicted motion model. The region matching technique attempts to track objects during occlusion by tracking segmented regions of uniform textures. When the content of an object is highly occluded by another object, regions are likely to be poorly matched with previously segment regions and mistracks easily occur. The hybrid technique relies on the fact that when such mistracks occur, the trajectory model-based tracker intervenes to provide a better predicted position. Because of the position uncertainty associated with both estimators, the correlation cost function is calculated within a small regions of 11×11 pixels around these predicted positions. This small search window also ensures that the predicted track remains within the vicinity of their expected positions. It will be shown in section 7.6 that both $\alpha - \beta$ and Kalman techniques perform similarly and that the Kalman filter is chosen as default trajectory model-based estimator. The object position is predicted by the hybrid estimator by

$$\mathbf{x}_t^* = \text{argmin}_{\mathbf{x}=\left\{\mathbf{x}_t^{*(k)}, \mathbf{x}_t^{*(r)}\right\}} \left\{ \frac{1}{\text{Card}(\mathcal{W})} \sum_{\mathbf{x}' \in \mathcal{W}} \left(T_\tau(\mathbf{x}') - I_t(\mathbf{x}+\mathbf{x}')\right)^2 \right\} \tag{7.30}$$

where $\mathbf{x}_t^{*(k)}$ is the position predicted by the Kalman filter (equation 7.14), $\mathbf{x}_t^{*(r)}$ is the position predicted by the region matching technique (equation 7.28) and $\mathcal{W}$ is the set of foreground pixels present in the template of the tracked object expressed in equation 7.23.

7.5 Summary of the tracking algorithm

Traditional object tracking is performed by using a trajectory model-based position predictor. Two trajectory model-based estimators are implemented in section 7.3.2, an $\alpha - \beta$ and Kalman filtering technique. Both produce highly accurate and successful tracking for each objects. However, when objects enter occlusion, where their corresponding foreground pixels are non-separable into individual blobs, motion model-based tracking is unreliable since the tracking is performed blindly *i.e.* irrespective of the object contents.

Three appearance model-based estimators are designed in section 7.3 to tackle the occlusion problem by using the grey-level information held by the tracked objects. The first estimator predicts object position during occlusion by using the traditional cross-correlation technique to minimise the sum of the square differences between grey-levels of the foreground pixels. The second estimator is the region matching technique which estimates positions by matching segmented uniform grey-level regions. The third technique, called the hybrid technique, is an hybrid technique of a trajectory model-based estimator and the region matching technique. It aims to use one technique when the other fail in tracking occluded objects and vice versa.

7.6 Evaluating tracking with stationary cameras

Six objects are tracked by the tracking framework described previously. Their tracks are evaluated in two sequences captured by two surveillance cameras in an outdoor environment. The two datasets are presented in section 7.6.1 with the ground-truth bounding boxes of each object. This section also explains how each ground-truth track is associated with the observed track by a best similarity measurement. Two metrics, defined in section 7.6.2, measure the tracking performance of each object. The first one measures the life-time of the ground-truth tracks during occlusion *i.e.* the time tracks are associated with the same 3D object, and the second metric measures how far the tracks are to their ground-truth positions. The results presented in section 7.6.3 show that the $\alpha - \beta$ and Kalman filters perform similarly in tracking objects. Although objects are moving consistently from frame to frame, these two trajectory model-based trackers often fail in tracking objects successfully in occlusion. The use of the three appearance model-based trackers (developed in section 7.4) improves the tracking results but perform differently depending on the occlusion density and the objects characteristics in occlusion.

7.6.1 Datasets and ground-truth

Tracks obtained after the data association process described in section 7.3 are evaluated against ground-truth data. The ground-truth foreground pixels are not provided, however an interface allowed the drawing of the bounding boxes of the ground-truth tracks. Figure 7.14 displays all the ground-truth positions of the 'PETS1' and 'PETS2' dataset on a single frame where one colour is assigned per track. The object characteristics are displayed in table 7.1. Both 'PETS' sequences contain the same six objects captured in the same environment but by two different cameras. Table 7.1 gives details about the total number of times each object remains in the scene and the number of times $N_O(i)$ they are in an occlusion where i is the object index in the array of objects.

Figure 7.14: Ground-truth object positions in the left: 'PETS1' and right: 'PETS2' sequence

Each ground-truth track is compared with all the observed tracked by the use of a membership function defined in equation 7.31. This membership measurement at a time t between

Object i	Description	Total number of appearances		Total number of times occluded: $N_O(i)$	
		'PETS1'	'PETS2'	'PETS1'	'PETS2'
1	grey cloth person	544	352	39	29
2	grey car	256	239	39	33
3	white van	320	490	68	44
4	dark cloth person	228	455	201	412
5	dark cloth person	216	452	211	297
6	white coat person	189	418	157	410

Table 7.1: Object characteristics in the 'PETS1' and 'PETS2' sequence

the i^{th} ground-truth $B_t^g(i)$ and the j^{th} observed track $B_t(j)$ is equivalent to as a similarity measurement in terms of position deviation $\Delta d_t'(i,j)$ and size difference $\Delta \mathcal{A}_t'(i,j)$:

$$\text{membership}_{\ t}(i,j) = \frac{1}{(1+\Delta d_t'(i,j))(1+\Delta \mathcal{A}_t'(i,j))} \tag{7.31}$$

where

$$\Delta d_t'(i,j) = \frac{\|\mathbf{x}_t^g(i) - \mathbf{x}_t(j)\|}{\sqrt{\mathcal{W}_t^g(i)^2 + \mathcal{H}_t^g(i)^2}} \tag{7.32}$$

$$\Delta \mathcal{A}_t'(i,j) = \frac{\Delta \mathcal{A}_t(i,j)}{\max(\mathcal{A}_t^g(i), \mathcal{A}_t(j))} \tag{7.33}$$

The distance error term $\Delta d'$ represents a normalised deviation between the observed and ground-truth track positions $\mathbf{x}_t$ and $\mathbf{x}_t^g$ respectively. The area error term $\Delta \mathcal{A}'$ represents a normalised difference between the bounding box area of the ground-truth track $\mathcal{A}^g$ and the observed track $\mathcal{A}$. Therefore large deviations in appearance or position from the ground-truth result in a weak membership value and vice versa. The maximum membership value is one and occurs when the two objects are totally identical.

Each i^{th} ground-truth track is associated with the j^{th} observed track giving the highest membership result $m_t(i)$ among the N_t observed tracks as defined by

$$m_t(i) = \begin{cases} \text{argmax}_{\ j\in[1:N_t]} \{\text{membership}_{\ t}(i,j)\} & \text{if } N_t \geq 0 \\ \text{NULL} & \text{else} \end{cases} \quad \forall\, i = [1:N_\tau] \tag{7.34}$$

The correspondences between ground-truth and observation are then evaluated by the use of two metrics described in the following section.

7.6.2 Metrics

The evaluation of the tracking algorithm is performed via two metrics on the set of correspondences between ground-truth and observed tracked established from equation 7.34. The first

metric $\Delta D(i)$ measures in equation 7.35 the mean deviation $d_t(i,j)$ between the i^{th} ground-truth object position and the position of the matching j^{th} observed object during the occlusion life-time denoted $\mathcal{F}_i$ *i.e.* $t \in \mathcal{F}_i$.

$$\Delta D(i) = \frac{1}{\mathcal{O}'(i)} \sum_{\substack{t \in \mathcal{F}_i \\ m_t(i) \neq \text{NULL}}} \Delta d(i, m_t(i)) \tag{7.35}$$

where $m_t(i)$ is the index of the blob giving the best match with the i^{th} object (see equation 7.34). The spatial deviation Δd is defined in equation 7.36 and the term $\mathcal{O}'(i)$ in equation 7.37 defines the number of times the i^{th} track find a match with an observed track during an occlusion.

$$\Delta d_t(i,j) = \|\mathbf{x}_t^g(i) - \mathbf{x}_t(j)\| \tag{7.36}$$

$$\mathcal{O}'(i) = \sum_{t \in \mathcal{F}_i} \begin{cases} 1 \text{ if } m_t(i) \neq \text{NULL} \\ 0 \text{ else} \end{cases} \leq \mathcal{O}(i) \tag{7.37}$$

The second metric $\Delta L(i)$ evaluates in equation 7.39 the average time the i^{th} ground-truth object survives with the same associated observed track in terms of frames number during occlusion. For example, if a ground-truth track always has for correspondence the same observed track during occlusion, then this track survives indefinitely and $\Delta L = \infty$. The minimum average time of survival is $\Delta L = 1$ which means that the correspondences of the ground-truth track are different at every captured frames.

$$\Delta L(i) = \frac{\text{Life} - \text{time of } i^{th} \text{ ground} - \text{truth object during occlusion}}{\text{Total number of different correspondences of this object}} \tag{7.38}$$

$$= \frac{\mathcal{O}(i)}{\sum_{t \in \mathcal{F}_i} \Delta l_t(i)} \tag{7.39}$$

where $\Delta l_t(i)$ takes a value of 1 if the track with which the i^{th} track is associated with is a different track from the previous time:

$$\Delta l_t(i) = \begin{cases} 1 \text{ if } m_{t-1}(i) \neq m_t(i) \\ 0 \text{ else} \end{cases} \tag{7.40}$$

7.6.3 Results

In the first experiment, the $\alpha - \beta$ and Kalman trajectory model-based estimators are compared and the results show that they perform similarly during occlusion. The second experiment evaluates the three appearance models trackers (described in section 7.4) and shows that tracking is improved but still unsatisfactory for successful tracking.

Kalman versus $\alpha - \beta$ filtering: finding a trajectory predictor

The two trajectory model-based trackers developed in section 7.3.2, the $\alpha - \beta$ and the Kalman filtering trackers, are evaluated in this section. Both estimators track successfully all objects when there is no occlusion occurrences in the 'PETS' datasets. Table 7.1 displays the measured

label change rate, or tracking life-time ΔL, and associated mean object deviations ΔD (see section 7.6.2) for the six ground-truth objects presented in section 7.6.1. The first three objects, objects 1, 2 and 3, are considered low occluded and of relative high dimensions compared to the three last objects.

Object i	$\alpha-\beta$ estimator				Kalman estimator			
	'PETS1'		'PETS2'		'PETS1'		'PETS2'	
	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$
1	6.55	19	31.81	14.5	14.99	7.8	13.45	∞
2	8.69	19	34.96	33	12.51	39	34.37	33
3	20.32	17	31.56	44	8.63	68	20.50	44
4	19.90	11.2	33.35	25.8	20.32	10.1	30.64	25.8
5	12.65	11.1	28.29	14.9	12.14	12.4	28.61	14.9
6	20.63	14.3	30.86	25.6	21.01	11.2	27.43	22.8

Table 7.2: Comparison results of the $\alpha-\beta$ and Kalman trackers during occlusion in the 'PETS1' and 'PETS2' sequence

The ΔD results never exceeds a relatively high value of 35 pixels which mean that the maximum object position error is of 35 pixels during occlusion for the three first objects. The life-time of a track is of minimum 19 frames and maximum ∞, therefore successful tracking rarely occurs. Generally, the first three objects are slightly better tracked by the Kalman filtering technique.

The $\alpha-\beta$ and Kalman tracker give similar tracking performances for the three last objects (object 4, 5 and 6) which are considered highly occluded and of small sizes compared to the three first objects. If an average of the results is to be performed for the ΔD and ΔL in table 7.2, the Kalman tracker would give slightly better results. Approximately, in the 'PETS1' sequence, the last three objects are positioned with a minimum deviation error of 12 pixels and maximum 21 pixels with a life-time of about 11 frames. In the 'PETS2' sequence, these objects positions are positioned with an approximate deviation of 27-30 with life-times between 15 and 26 frames.

Comparing the results of the Kalman estimator between the three first and three last objects, the low occluded objects (the three first objects) are in general better tracked than the highly occluded objects (the three last objects). The Kalman estimator seems to be slightly better suited for tracking objects than the $\alpha-\beta$ estimator and is compared in the next experiments with three appearance model-based trackers.

Trajectory versus appearance model-based tracking

The Kalman filter estimates position blindly when occlusion occurs whereas appearance models trackers use local grey-level information to attempt better tracking. The three appearance

model-based trackers described in section 7.4 *i.e.* the correlation, region matching and hybrid trackers are evaluated and compared to the Kalman tracker in table 7.3 for the 'PETS1' sequence and in table 7.4 for the 'PETS2' sequence.

Object i	Estimator							
	Kalman		Correlation		Region matching		Hybrid	
	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$
1	14.99	7.8	4.60	39	21.18	7.8	15.33	9.8
2	12.51	39	8.48	39	16.94	5.6	15.34	39
3	8.63	68	7.44	68	21.27	22.7	14.26	22.7
4	20.32	10.1	26.24	10.1	22.39	10.1	21.66	9.6
5	12.14	12.4	17.60	8.1	11.16	14.1	10.74	11.7
6	21.01	11.2	24.07	8.3	21.79	12.1	18.19	13.1

Table 7.3: Evaluation results during occlusion for the 'PETS1' sequence

Object i	Estimator							
	Kalman		Correlation		Region matching		Hybrid	
	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$	$\Delta D(i)$	$\Delta L(i)$
1	13.45	∞	27.09	∞	35.80	29	27.24	∞
2	34.37	33	18.62	33	27.52	33	25.37	33
3	20.50	44	25.90	44	23.00	44	22.29	44
4	30.64	25.8	24.41	27.5	33.23	21.7	31.28	27.5
5	28.61	14.9	22.73	14.9	35.38	14.1	29.07	15.6
6	27.43	22.8	33.01	21.6	29.97	24.1	29.05	27.3

Table 7.4: Evaluation results during occlusion for the 'PETS2' sequence

The correlation estimator is the most accurate technique for tracking the first three objects, object 1, 2 and 3, in the 'PETS1' sequence which are categorised as low occluded objects with significant dimensions. For the 'PETS2' results, only the first and the third objects are best tracked by the Kalman technique. Hence, the correlation technique seems an appropriate technique for tracking these relatively large and low occluded objects.

The last three objects, object 4, 5 and 6, are categorised as highly occluded with poor contrasted grey-level content in small object dimensions. Because the results of position error ΔD and life-times ΔL are relatively similar for the four different trackers, their performances are ranked in function of their life-times results only in table 7.5. A technique is ranked first if its corresponding ΔL is the highest compared to the other techniques for a given sequence and vice versa. In this table, the Kalman, correlation, region-matching and hybrid tracker are represented by *Klm*, *Corr*, *Rm* and *Hyb* respectively.

Object	Rank of performance							
	'PETS1' sequence				'PETS2' sequence			
	1^{st}	2^{nd}	3^{d}	4^{th}	1^{st}	2^{nd}	3^{d}	4^{th}
4	*Klm*	*Rm*	*Corr*	*Hyb*	*Corr*	*Hyb*	*Klm*	*Rm*
5	*Rm*	*Klm*	*Hyb*	*Corr*	*Hyb*	*Corr*	*Klm*	*Rm*
6	*Hyb*	*Rm*	*Klm*	*Corr*	*Hyb*	*Rm*	*Klm*	*Corr*

Table 7.5: Evaluation results during occlusion of the 'PETS' sequences

Table 7.5 shows that the estimators have varying ranks with the different objects and sequences. The region-matching techniques has a good ranking with the 'PETS1' sequence but it has a worse ranking in the second sequence whereas the opposite is observed for the hybrid technique. An average of the life-times in the first two tables gives the following ranking: the region-matching is 1st, the hybrid is 2nd, the Kalman filter is 3rd and the correlation is last for the 'PETS1' sequence. The hybrid is 1st, the correlation is 2nd, the Kalman filter is once again 3rd and the region-matching is last for the 'PETS2' sequence. The hybrid technique is ranked 1st and 2nd in both sequences and seems to be a slighter better tracker than the others for these three highly occluded objects whereas correlation gave the best tracking results for the low occluded objects.

The first three objects displaying larger dimensions with more contrasted grey-level content and less occluded than the last three objects are better tracked than these latter objects. The correlation tracker gave the best results for the low occlusion case whereas the hybrid tracker gave the best results for the higher occluded case. However, all of these six objects are mistracked during occlusion, therefore additional information is necessary for complete tracking.

7.6.4 Discussion

The performances of the two trajectory model-based trackers, the $\alpha - \beta$ and the Kalman technique, are very similar. Although they successfully track objects, they fail to track objects during occlusion scenarios where information about object positions is lost, or more precisely hidden. The Kalman filter technique has a slight disadvantage over the $\alpha - \beta$ filter since it requires several frames before tracking accurately fast accelerating objects when they appear for the first time in the scene. However, this disadvantage does not prevent the Kalman estimator to perform slightly better than the $\alpha - \beta$ technique.

The Kalman tracker is then compared to three appearance model-based trackers *i.e.* the correlation, region matching and hybrid estimator which attempt to recover the hidden object positions during tracking by using the grey-level content of these objects. The comparison results showed that the trackers performances vary with the nature of the objects, more specifically, their occlusion densities, their dimensions and their pixel grey-level contents. When objects are

low occluded, the correlation technique was the most performant technique. However, when occlusion density of small and poorly contrasted objects increases, all the trackers gave similar and poor performances with slightly better results with the hybrid technique.

The occlusion problem remains a sensitive issue in tracking objects but can be properly addressed when additional cues are integrated in the appearance models. Cues such as colour information, shape models and depth of objects would for example improve greatly the tracking results.

7.7 Conclusion

A multiple tracking framework is developed in this chapter to attempt to address the problem of occlusion which cause most tracking algorithms of the literature to fail to track objects successfully. This problem occurs when 3D foreground objects in a static scene interact in a way that their projected images overlap on the 2D plane of the camera. Before being tracked, foreground objects are segmented from the background scene from the two steps procedure described in section 7.2. First, a statistical background reference frame is built over time by the Stauffer and Grimson's multi Gaussian technique. The foreground segmentation is performed by the multi-variate technique was shown noisy but accurate enough to allow in the second step the grouping of the foreground pixels into individual clusters, referred to as blobs, by a projected histogram technique.

The occlusion problem causes the foreground object segmentation technique to fail to separate the groups of merging foreground pixels into individual blobs during occlusion. Recovering the objects positions lost during occlusion is the aim of the data association module described in section 7.3. Two trajectory model-based trackers, a zero-acceleration $\alpha - \beta$ and Kalman technique, successfully track objects when there is no occlusion. However, when occlusion occurs, they easily mistrack objects as they blindly track them according only to motion models.

The Kalman tracker is shown to give slightly better results than the $\alpha - \beta$ tracker and is then compared to three appearance model-based tracker which attempts to recover the lost information of the objects during occlusion. The first technique is the correlation tracker which uses a least-squares approach to minimise grey-level similitude between objects to predict the best position. This estimator is sensitive to noise and performs relatively well only for low-occluded objects with significant and distinctive grey-level distributions. The second technique is the region-matching tracker which is designed to match segmented grey-level distributions. It is design to overcome the noise problem of the correlation technique when objects are significantly occluded by other objects. This technique is not performant in segmenting accurately the different regions of small captured objects and, in general, it does not give better tracking results than the correlation technique. The third technique is the hybrid tracker, designed to switch between the Kalman position estimates and the region-matching estimates. These three techniques gave similar performance and all fail to accurately estimates although a slightly

better performance was obtained with the correlation tracker for low-occluded objects and by the hybrid tracker for highly-occluded objects. Nevertheless, although the amount of evaluation data is too poor, similar observations and results are expected to occur with more intensive evaluation and additional information such as colour, shape and 3D object positions is required for successful tracking.

Chapter 8

Detecting objects with pan and tilt cameras

Detecting foreground pixels when PTZ cameras are moving cannot be performed as reliably as when cameras are stationary as demonstrated in the previous chapter where a reliable modelling of the background is possible by a multi-variate technique. In the first application described in this chapter, cameras are allowed to move in any direction and only the focal length remains fixed. The first section reviews the main techniques developed in the literature to perform motion estimation in order to compensate for *global motion* induced by the camera's motions. Motion compensated background allows the segmentation of the foreground motions, also referred to as the *local motions*.

An overview of the foreground objects detection algorithm is given in the second section which describes how moving objects are detected in section 8.4 using thresholding between the captured frame and a motion compensated background reference frame. The results, displayed for three sequences in section 8.6, show a relatively accurate segmentation of the moving objects but with high associated noise. A foreground edge-based segmentation is also developed in section 8.4 which allows a significant filtering of the noisy moving objects in the frame. Before the entire segmentation process is repeated, the background reference frame is updated using an averaging grey-level approach described in section 8.5.

8.1 Review

Foreground objects captured by surveillance moving PTZ cameras are mostly detected when the global motion associated with the static object of the background scenes is estimated. Global motion estimation is an image registration technique which aims to find a general relationship or transformation between images. The techniques and applications of image registration are reviewed in the first section and the second section concentrates on reviewing the techniques

used to segment and track objects captured by moving PTZ cameras.

8.1.1 Image registration

Image registration is an important problem in computer vision [24], from remote sensing applications [2, 142] to medical image analysis applications [7, 50, 51]. The key step of image registration, also known as *spatial normalisation* (SPM), is to find the spatial transformations of objects such that features between two or more images taken at different times, from different sensors or from different viewpoints, give maximum similarities.

The relationship between images may correspond to a single global transformation, as in the case of different views of a static planar surface for example, or to a spatially varying transformation field, possibly including discontinuities, as in the case of a non-static scene. In either case, the registration problem is a difficult one and often complicated by such things as occlusions, ambiguous matches and the presence of observation noise and distortion. The task can be eased by using knowledge of the imaging process to constrain the class of transformation field being estimated. The simplest example is to assume that the motion field is translational. This assumption is sometimes a reasonable assumption in remote sensing areas that basically involve the identification of many control points in the images. The increased volume of satellite images has reinforced the need for automatic image registration methods. A more general approach is to assume that the images are related by a six parameter affine transformation, corresponding to dilation, rotation, shear and translation. A wide variety of registration techniques have been developed for different applications as the ones previously mentioned. These techniques may be classified into:

- correlation methods
- gradient-based methods
- Fourier methods
- landmark mapping and,
- elastic model-based matching

The correlation methods are generally limited to registration problems [142] in which images are misaligned by only a small rigid transformation. In addition, the peak of the correlation may not be clearly discernible in the presence of noise. Fourier methods [53, 66, 86] are the frequency domain equivalent of the correlation methods. Fourier methods make use of the translational property of the Fourier transform and search for the optimal spectral match between two images. However, Fourier matching methods are also limited to registration problems with small rigid transformations. If there exists spatially local variation, then both the correlation methods and the Fourier methods would fail. Gradient-based methods use the grey-level changes in the spatio-temporal domain to measure motion (see chapter 6). Moreover, such gradient-based techniques

allow for the implementation of a motion model which account for the small rigid deformation problem previously mentioned and a better matching is usually obtained. Because computing optical flow is an expensive process, it is only computed at sparse feature points as achieved by Fusiello *et al* [56] which then reject feature outliers by using robust statistics.

For cases of unknown misalignment type, landmark mapping and elastic model-based matching techniques [2, 50, 51] may be used to tackle the registration problem. Landmark mapping techniques extract feature points from a reference image and a target image, respectively, and then apply a piecewise interpolation to compute a transformation for mapping the feature point sets from the reference image to the target image. When such transformations are not applicable to the selected features between a set of frames, the alternative method of greedy search is deployed. The widely used RANSAC technique is a robust technique that matches the best features and iteratively rejects outliers as used by Matas *et al* [90] and Clarke *et al* [35].

Elastic model-based matching techniques are similar to feature-based techniques. Altinalev *et al* [2] detect object contours as features and are registered with cross correlation matching to compute scale and translation. Ferrant *et al* [50, 51] generates a bio-medical model for the cortical surface of MR (Magnetic Resonance) images as elastic bodies. When small strains occur on the surfaces, they can be modelled as linear elastics and are represented as tetrahedral mesh.

Surveillance image registration focuses on registering pair of successive images captured as the camera moves. The motion associated with the dominant background scene is generally modelled by affine transformations. The registration of such motion is equivalent to motion detection and the foreground objects are detected via. *motion compensation* defined below.

Motion compensation

Registering the complex motion associated with the stationary objects captured by a moving camera is achieved by a motion compensation technique. The motion due to object movement is referred to as *local motion* or *object motion*, and the motion due to camera motion or operation is referred to as *global motion*, as it represents the majority of the entire motion present in the scene. Motion compensation is achieved by a technique known as *global motion estimation* or *GME* (see chapter 5), and the process of motion compensation is referred to as *global motion compensation* or *GMC*. These techniques consist mainly in rejecting the local motions which contaminate the GMC.

GMC is an important tool for a variety of video processing applications, including for instance segmentation and coding. The basic idea is that a part of the visible 2D motion within video sequences is caused by camera operations. A common approach is to model this global motion by a parametric 2D model which allows the subsequent segmentation of the local motions caused by foreground objects. Tom and Katsaggelos [130] for example use the additional information provided by motion compensation to improve a multiple input algorithm [79] in order to enhance image resolution.

Mosaicking

Global motion compensation allows the updating of the background reference frame by registering several background frames. This is performed either on the part of the background covered by the camera's view as it moves or on a large mosaic of the background, also known as *sprite*. Mosaicking is the automatic alignment of multiple images into larger aggregates onto a common reference plane, [126, 143], and is often applied to video coding applications such as in the MPEG4 technique [100]. Babu [9], on another hand, uses the motion vector provided by the MPEG compression technique to generate the sprite. Farin *et al* in [48] improve the resolution of the sprite by coding the background as several separate sprites improve the reduction of data held in the background sprite.

8.1.2 Tracking in PTZ imagery

Object tracking algorithms are typically based on the detection of a particular cue, most commonly colours, edges, shapes, textures or feature templates. The application of object tracking in video-surveillance has a large range of purposes, from traffic monitoring [83] to human activity understanding [42].

Koller *et al* [83] use parameterised 3D models to model the shape of vehicles and an associated motion model to track vehicles in stationary scenes. Edge segments are detected within the scene and are matched to represent the best fitting vehicle model. Similar work [127] track objects using multi-cues features such as colour, edge and texture. Araki *et al* [6] track object contours in moving cameras after motion compensation of the background. The contours are represented by active parameterised contour models referred to as *snakes*. The background affine motion model is estimated via the motion estimates of object features given by a block-matcher. The LMedS (Least Median of Squares) robust statistical technique separates the foreground features from the background features.

In human activity understanding, colour has been greatly used in machine-based vision systems for tasks such as segmentation and recognition. Colour cues have been shown to offer several significant advantages over geometric information for certain tasks in visual perception, such as robustness under partial occlusion, rotation in depth, scale changes and resolution changes. Furthermore, colour processing can often utilise efficient algorithms yielding to real-time performance on standard hardware. McKenna *et al* [91] adopts a statistical approach in which colour distributions are modelled over time by the mean of adaptive colour mixture models. These dynamic models estimate an object's colour distribution on-line and adapt to accomodate changes in the viewing conditions. They track objects robustly and in real-time under variations in illumination, viewing geometry and under varying camera parameters.

Foresti and Micheloni [52] deploy the Tomasi-Kanade technique to detect and track Shi and Tomasi-based features [115]. The best detected features are registered using a robust rejection rule and allow for a translational motion to be estimated. Foreground pixels are detected by

thresholding change detection and are grouped together for object analysis. Optical flow motion vectors are estimated at the location of sparse Shi and Tomasi-based features in the work of Fusiello *et al* [56].

Multi-cue integration has been exploited extensively in feature-based tracking applications [47], such as in people tracking [40, 120]. Other authors integrate several cameras for better reasoning occlusion of multiple walking persons [42]. Darrell *et al* [40] use a stereo algorithm to compute the different depth of the objects in a face tracker. The face is detected and tracked by the use of skin colour segmentation and a trained neural network for the detection of face patterns. Murray and Basu [94] also use a gradient-based optical flow technique. They aim to compensate the global motion of the background objects on a sparse set of points located on the image edges. Moving pixels are segmented by an adaptive filter on the pixel differences between the motion compensated captured frame and the previously captured frame. The use of morphological filters on the edge map filters out the moving regions into the right moving foreground objects of the new frame.

8.2 Overview of the moving objects detection algorithm

Foreground objects are detected in this chapter by a three step algorithm, as shown in figure 8.1. First, the global motion of the background scene captured between two different times is estimated in section 161. Second, the three modules of the moving segmentation step segment and filter in section 8.4 the moving clusters of pixels, also referred to as *blobs*, corresponding to the foreground objects. In the first module of the segmentation process, module 1 in figure 8.1, the reference background frame is predicted at capture time for comparison with the captured frame. Moving objects are detected into blobs using a threshold on a contrast measurement. However, the blob segmentation technique is noisy at highly contrasted pixels and low grey-level pixels. A moving edge segmentation technique is then implemented in the second module, module 2 in figure 8.1, to segment moving edge blobs. The third module, module 3 in figure 8.1, combines the information of the two sets of blobs to filter out the noisy blobs while keeping the same segmentation accuracy. Finally, an update of local background pixels is performed in the third step before the next captured objects can be segmented, as described in section 8.5.

8.3 Global motion estimation

The robust and hierarchical optical flow technique developed in chapter 6 models the background scene as a 3D surface in which relative motion of the camera is modelled by a single planar motion model. This gradient-based technique is constrained by a large number of approximately 2,500 edge-based pixels as neighbouring pixels of the background scene. The results of chapter 6 showed that global motion is estimated with relatively high accuracy when there are no occluding foreground objects. The robust statistical approach first used with the local optical

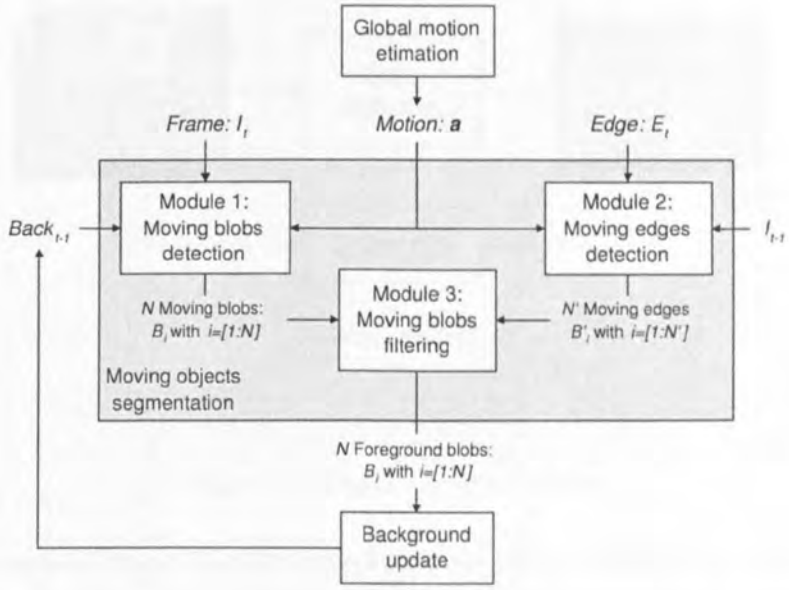

Figure 8.1: Moving objects segmentation strategy

flow estimator in chapter 4 showed neighbourhood can be contaminated with motion noise up to 30%. In this application, it is assumed that the percentage of the number of edges belonging to the foreground objects is not high enough to perturb the global motion estimation of the background scene. It is also assumed that the maximum motion magnitude in a pan and tilt sequence is below the maximum motion that the hierarchical approach can cope with.

The global motion modelled by a planar motion model $\mathbf{a}$ is estimated between frame I_{t-1} previously captured at time $t-1$ and frame I_t captured at current time t as illustrated in the diagram of figure 8.2. The edge sampling module in this figure samples 2,500 edge-based neighbouring pixels, represented by the mask Mask_t, among the total edge pixels which are detected by a simple thresholding on Sobel kernel responses as explained in detail in section 6.1.2. The optical flow module represents the robust and hierarchical global motion estimator. The 8-parameter motion vector $\mathbf{a}$ gives the displacement at any pixel location on the background scene according to a motion matrix X as reported in chapter 3:

$$\Delta\mathbf{x} = \Delta\mathbf{x}(\mathbf{x}, \mathbf{a}) = X(\mathbf{x})\mathbf{a} \tag{8.1}$$

with

$$X(\mathbf{x}) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{bmatrix} \tag{8.2}$$

8.4 Segmenting moving objects

Foreground pixels are detected in the first section 8.4.1 by a thresholding operation on the grey-level differences between the pixels of the motion compensated background frame and the pixels

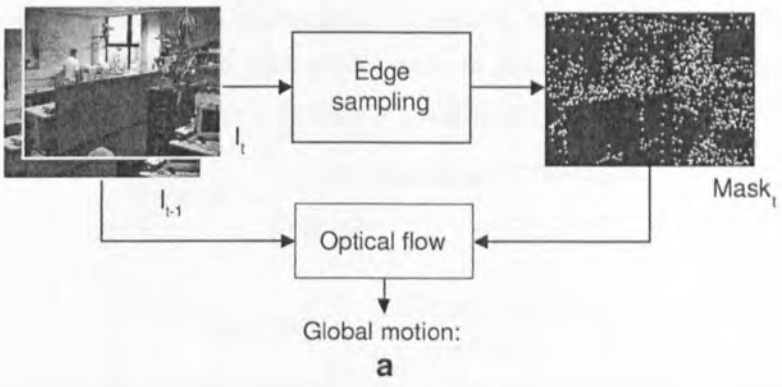

Figure 8.2: Global motion estimation

of the new captured frame. These foreground pixels are then grouped into moving blobs by the connected-component technique. However, the segmentation process is noisy, producing noisy blobs which are then filtered in the third section by the use of moving edge information detected in the second section.

8.4.1 Module 1 - Moving objects detection

Pixels of the new captured frame which do not belong to the background as segmented are foreground pixels. The background frame $Back_t$ is not available at capture time t and is instead predicted *i.e.* $\widetilde{Back}_t$ from the previous time $t-1$ as follows

$$\widetilde{Back}_t(\mathbf{x}) = Back_{t-1}(\mathbf{x} - \Delta\mathbf{x}) \tag{8.3}$$

where $\Delta\mathbf{x}$ is the optical flow at pixel location $\mathbf{x}$, given the motion parameter vector $\mathbf{a}$ estimated by the global motion estimator of section 8.3: see equation 8.1. Because of problems with image registration accuracy, the multi-Gaussian technique used in the previous chapter to build the stationary background frame and segment the foreground pixels, cannot be used in this pan and tilt application.

The Stauffer and Grimson technique would the most appropriate technique to construct a reliable background. However, it requires an accurate positioning of the background pixels. Inaccuracy often occurs in image registration applications as observed in chapter 6 where a single planar surface is forced to fit complexed background scene. The dynamic thresholding provided by the Stauffer and Grimson technique is hence not available.

The background frame is instead built by a temporal averaging approach described in the next section 8.5. The traditional threshold on the absolute grey-level difference between the background and current frame seems to be the obvious choice. However, it was observed that the threshold varies greatly with the choice of data. The contrast thresholding technique described by equation 8.5 was preferred. The threshold was empirically fixed to 15% for all datasets. However, this technique has the main disadvantage of sensitivity in low grey-level regions of the

background where small variations in captured grey-level from the background can be easily considered as foreground, hence noise occurs easily in these regions. Foreground pixels are segmented by the following contrast thresholding operation:

$$Fore_t(\mathbf{x}) = \begin{cases} 1 & \text{if Contrast}(\mathbf{x}) < \text{Threshold} \\ 0 & \text{else} \end{cases} \tag{8.4}$$

where

$$\text{Contrast}(\mathbf{x}) = \frac{|\widetilde{Back}_t(\mathbf{x}) - I_t(\mathbf{x})|}{\widetilde{Back}_t(\mathbf{x})} \tag{8.5}$$

As explained earlier, although the gradient-based global motion estimator provides relatively accurate results, a 3D surface is modelling a complex background scene in terms of content and the estimated planar motion model cannot give a perfect registration of all the pixels in the frame. Therefore, foreground noise is also likely to occur in at highly contrasted pixels *i.e.* the edges.

The foreground segmentation technique is illustrated in the diagram of figure 8.3 where the |Diff| represents the absolute difference operation between the predicted background and the captured frame. The thresholding module operation segments the foreground pixels, represented in dark in the foreground frame $Fore_t$, which are then grouped into a set of N blobs B_i with $i = [1 : N]$ by the connected-component technique:

$$\mathcal{B} = \{B_i, i = [1 : N]\} \tag{8.6}$$

The foreground segmentation technique is generally noisy and produces in general many small noisy blobs at edges in dark regions of an image. A filtering is therefore performed on the moving blobs where any moving blobs with a lateral dimension lower than a threshold of 7 pixels are eliminated. In addition, another filtering process is performed on the moving blobs in section 8.4.3 by using the moving edge information technique described in the next section.

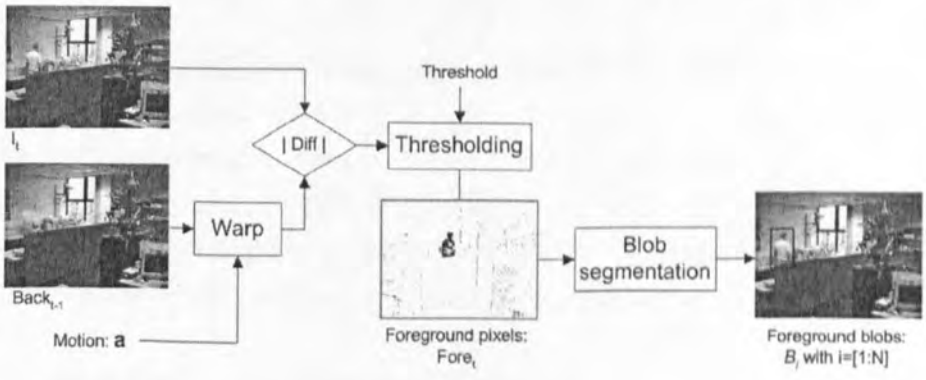

Figure 8.3: Module 1: moving objects detection

8.4.2 Module 2 - Moving edge detection

The foreground blob detection technique described in the first section produces noisy blobs in low grey-level regions and at high contrast pixels of the background frame. The robust statistical regressor used in chapter 4, *i.e.* the median regressor, is here used to eliminate edge-based outliers that do not belong to the background scene. The foreground edges are also referred to as the *moving edges.* It is assumed the total number of foreground edges represent a small percentage of the total number of edges in a capture frame I_t at time t. A pixel at location $\mathbf{x}$ is segmented as moving edge $E'_t(\mathbf{x}) = 1$ if it is segmented as foreground edge *i.e.* $\mathbf{x} \in \mathcal{E}'_t$

$$E'_t(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in \mathcal{E}'_t \\ 0 \text{ else} \end{cases} \tag{8.7}$$

where $\mathcal{E}'_t$ is the set of edge pixels segmented robustly as foreground:

$$\mathcal{E}'_t = \{\mathbf{x}; \mathbf{x} \in \mathcal{E}_t,\ |I_t(\mathbf{x}) - I_{t-1}(\mathbf{x} - \Delta\mathbf{x})| > 3\,\sigma^*\} \tag{8.8}$$

The term $\mathcal{E}_t$, defined in equation 8.9, represents the set of all the edge pixels segmented by the technique described in section 8.3, and σ^* is the median robust scale applied on the edge pixels as defined in equation 8.10.

$$\mathcal{E}_t = \{\mathbf{x}, E(\mathbf{x}) = 1\} \tag{8.9}$$

$$\sigma^* \approx 1.4826 \times \text{Median}_{\mathbf{x} \in \mathcal{E}_t} (|I_t(\mathbf{x}) - I_{t-1}(\mathbf{x} - \Delta\mathbf{x})|) \tag{8.10}$$

where E is the edge map of the same size of the original frame and a pixel located at location $\mathbf{x}$ is an edge if $E(\mathbf{x}) = 1$ and is not an edge if $E(\mathbf{x}) = 0$. The optical flow vector $\Delta\mathbf{x}$ is calculated from the previously estimated global motion model vector $\mathbf{a}$ in equation 8.1. The connected-component technique groups the moving edge pixels into a set $\mathcal{B}'$ of N' blobs B'_i with $i = [1 : N']$:

$$\mathcal{B}' = \{B'_i, i = [1 : N']\} \tag{8.11}$$

and the entire moving edge detection process is illustrated in figure 8.4.

In this figure 8.4, the 'Warp' operation is the motion compensation of the term I_{t-1} in equation 8.8. The 'Diff' operation is the difference between I_t and I_{t-1} in the same equation. The 'robust scale' is the calculation of σ^* named 'Threshold' (equation 8.8) and the connected-component operation is represented by the 'Blob segmentation' module. The segmented blobs of the moving edges are then used to filter the moving edges previously segmented and this is described below.

8.4.3 Module 3 - Moving objects filtering

The technique described by the first module of the moving objects detector extracts relatively accurately N moving objects B_i characterised by bounding boxes but the detection is noisy and unwanted noisy objects appear in dark regions and at the edges. In order to eliminate these noisy blobs, a simple filtering rule is applied with the use of the moving edges extracted by the

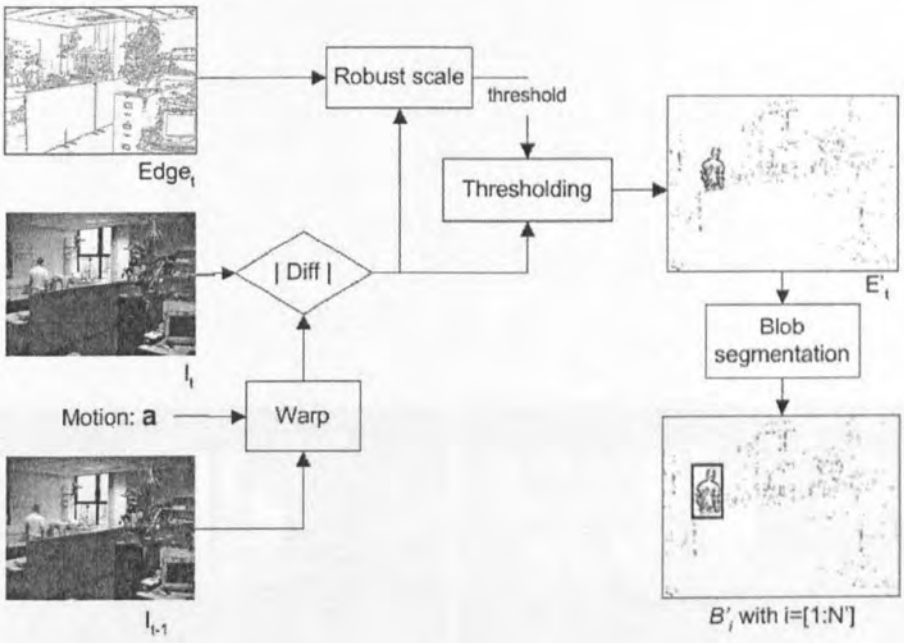

Figure 8.4: Module 2: moving edge detection

second module described in the previous section. All foreground blobs which do not intersect any moving edges blobs are eliminated. This greatly reduces the noise occurring at edges and in the dark regions of the background where edges do not generally appear. In addition, the remaining moving blobs have their bounding boxes updated with all the bounding boxes of the moving edges blobs they intersect, as defined in equation 8.13 creating a new set of moving foreground objects $\mathcal{B}$:

$$\mathcal{B} = \{B_i, i = [1 : N]\} \tag{8.12}$$

where

$$B_i = (B_i \cap B'_1) \cup (B_i \cap B'_2) \cup \ldots \cup (B_i \cap B'_{N'}) \tag{8.13}$$

An example of detected moving objects and moving edges are shown in the left and right figure of 8.5 respectively. In this example, a single object corresponding to a walking person is moving in the scene. This object is well segmented into the dark grey-level rectangle with 5 smaller noisy blobs drawn in white grey-level. The moving edge detection technique does not segment the walking person as well as with the moving objects detection technique but less noise is obtained. Moreover, the noisy blobs of both techniques do not appear at the same location (they do not overlap) therefore the noisy moving objects are eliminated during the filtering process. The blob of the walking person intersects with different blobs of the edges of this object which is therefore conserved as displayed in figure 8.6.

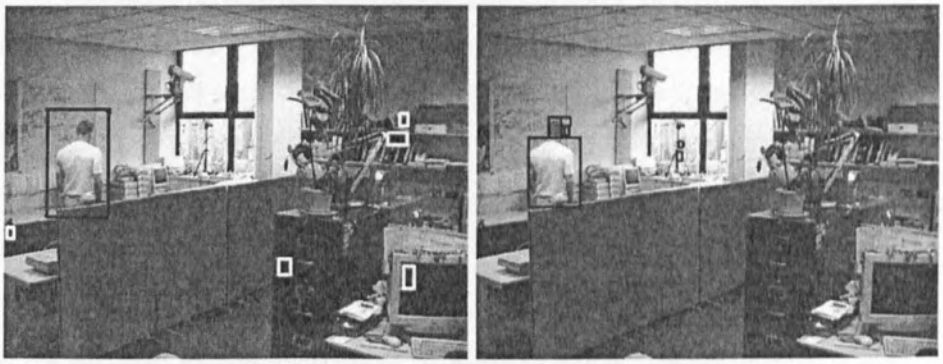

Figure 8.5: Left: moving objects and right: moving edge blobs

Figure 8.6: Filtered moving objects

8.5 Updating the reference background frame

In this application, no mosaic of the background is constructed but instead a local and reliable background is updated at every captured frame. The background frame needs to be updated with the content of the captured frame before being predicted and compared with the next captured frame by the moving objects detection technique described in section 8.4.1. A temporal averaging approach is developed in this section to update the grey-levels of the background according to the segmented and filtered moving objects (see section 8.4.3).

Each background pixel is associated an occurrence counter M_t which tells how many times a pixel represented a background object from the first time of capture until current time t. The counter $M_t(\mathbf{x})$ at location $\mathbf{x}$ is either decreased by one (provided it never goes negative) if it is detected as foreground *i.e.* $Fore_t(\mathbf{x}) = 1$ (see equation 8.4) or it is increased by 1 if it belongs to the background *i.e.* $Fore_t(\mathbf{x}) = 0$ provided that the counter never exceeds a threshold set empirically to 100:

$$M_t(\mathbf{x}) = \begin{cases} \min(\widetilde{M}_t(\mathbf{x}) + 1,\ 100) & \text{if } Fore_t(\mathbf{x}) = 0 \\ \max(\widetilde{M}_t(\mathbf{x}) - 1,\ 0) & \text{else} \end{cases} \tag{8.14}$$

where the predicted counter $\widetilde{M}_t$ is calculated similarly to equation 8.3:

$$\widetilde{M}_t(\mathbf{x}) = M_{t-1}(\mathbf{x} - \Delta\mathbf{x}) \tag{8.15}$$

The grey-levels of the background are updated according to their positions respective to the moving blobs bounding boxes as explained in the following two sections.

Foreground pixels

The grey-level of every background pixel corresponding to a pixel detected as foreground and located inside a moving object blob, *i.e.* $\mathbf{x} \in \mathcal{B}$ (defined in equation 8.12), is not updated. A background pixel occluded by a foreground pixel is simply updated by the background grey-level $\widetilde{Back}_t(\mathbf{x})$ predicted in equation 8.3 from the previous time of capture $t-1$:

$$Back_t(\mathbf{x}) = \widetilde{Back}_t(\mathbf{x}) \text{ if } \mathbf{x} \in \mathcal{B} \tag{8.16}$$

Background pixels

All background pixels non-occluded by a foreground object and which do not belong to any moving objects have their grey-levels updated with the new captured grey-level. The foreground detection is shown noisy and any background pixel inside a foreground object bounding box is likely to be foreground and update errors are shown to propagate as the same objects are being captured. Therefore, if a background pixel is located within one of the moving blobs bounding box, its grey-level is not only updated with the new grey-level but also with the

previous predicted background grey-level. The updates are expressed in the following equation

$$Back_t(\mathbf{x}) = \begin{cases} Back'_t(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{B} \\ I_t(\mathbf{x}) & \text{else} \end{cases} \tag{8.17}$$

where

$$Back'_t(\mathbf{x}) = \frac{\left(\widetilde{M_t}(\mathbf{x}) \times \widetilde{Back_t}(\mathbf{x})\right) + I_t(\mathbf{x})}{\widetilde{M_t}(\mathbf{x}) + 1} \tag{8.18}$$

A background frame example is displayed in the right figure of 8.7 and the corresponding captured frame in the left figure. A walking person walks within the scene and is being tracked by a moving camera. It can be seen in the background window of the moving object the blurring effect introduced by equations 8.16 and 8.17. Ground-truth data are provided in the result section 8.6 and allow the evaluation of the foreground object segmentation technique previously described in section 8.4 for various surveillance datasets.

Figure 8.7: Left: input frame with walking person and right: the corresponding background frame without the person

8.6 Evaluating foreground objects segmentation

This section presents the evaluation results of the moving object segmented by the algorithm described in section 8.4. Three surveillance datasets are used for the evaluation and are presented in the first section with their object ground-truth bounding boxes. The accuracy of the detected blobs are evaluated against the ground-truth data by the use of three metrics defined in the second section.

The segmentation accuracy is directly dependent on the accuracy of the global motion estimation that allow image registration of pair of images in a sequence, as described in section 8.3, and on the accuracy with which the local background is updated in section 8.5. The results show that accurate but noisy segmentation of the moving foreground objects is obtained. The filtering process of the moving blobs, which uses moving edge information, is significantly improved but remains noisy depending on the nature of the sequence *e.g.* the background and foreground objects content or the dynamic of the captured scenes.

8.6.1 Datasets and ground-truth

Three datasets are used for the evaluation of the three motion detectors of section 5.4.1 *i.e.* the moving objects detector, the moving edges and the filtered moving objects detector. An example frame of these datasets are displayed in figure 8.8 which are called the 'dirc', 'ptzets' and the 'lab' sequence.

Pan and tilt cameras are used to capture these sequences and do not perform any zooming operations, hence the focal length remains fixed at all times. The 'lab' and 'dirc' sequences contain 400 and 950 frames respectively and are captured indoors with the same surveillance camera using two different focal lengths. These two sequences capture a single foreground object: a walking person. The third sequence, the 'ptzets' sequence, contains 1000 artificially captured frames created by selecting sub-images from large input images captured by a stationary camera. The global motion between each pair of frame is therefore translational, unlike the two other sequences. The original sequence from which the 'ptzets' sequence is created is the 'PETS1' sequence used to evaluate multiple object tracking algorithm in chapter 7. Ground-truth bounding boxes of the moving objects in the three sequences are provided by a drawing interface algorithm against which the detected objects areas are compared with the use of three metrics defined in the next section.

Figure 8.8: Frame example of the left: 'dirc', middle: 'ptzets' and right: 'lab' sequence

8.6.2 Metrics

The three metrics defined below compare the N segmented foreground blobs B_i where $i = [1 : N]$ with respective bounding box area A_i against the N^g ground-truth blobs B_j^g where $j = [1 : N^g]$ with respective area A_j^g. The first metric, denoted E_c, evaluates in equation 8.19 the system accuracy in terms of a mean percentage of covered ground-truth object areas for a given sequence composed of N_F pairs of frames indexed by $t = [1 : N_F]$:

$$\begin{aligned} E_c &= \text{mean}\left\{\frac{\text{total ground} - \text{truth area intersecting detected blobs areas}}{\text{total ground} - \text{truth area}}\right\} \\ &= \frac{1}{N_F}\sum_{t=1}^{N_F}\left(\frac{\sum_{i=1}^{N}\sum_{j=1}^{N^g} A_i \cap A_j^g}{\sum_{j=1}^{N^g} A_j^g}\right) \end{aligned} \tag{8.19}$$

The second metric, denoted E_n, evaluates in equation 8.20 the noise level in the system by calculating the mean observed areas non-covering the ground-truth object areas *i.e.* the noisy

areas. The noise level calculation is performed relative to the total number of pixels in each frame, denoted Card($\mathcal{F}$) where $\mathcal{F}$ is the set of pixels within each frame.

$$E_n = \text{mean}\left\{\frac{\text{detected blobs areas non} - \text{intersecting withground} - \text{truth area}}{\text{total frame area}}\right\} = \frac{1}{N_F}\sum_{t=1}^{N_F}\left(\frac{\sum_{i=1}^{N}\sum_{j=1}^{N^g} A_i \cap \bar{A}_j^g}{\text{Card}(\mathcal{F})}\right) \quad (8.20)$$

where $\bar{A}$ is the entire frame area without the area covered by A. Finally, the third metric, denoted ΔN, evaluates in equation 8.21 the number of noisy objects obtained by the motion detectors which is calculated by the mean of the absolute difference between the number of observed and ground-truth blobs:

$$\Delta N = \frac{1}{N_F}\sum_{t=1}^{N_F}(|N^g - N|) \quad (8.21)$$

Therefore, according to these three previous metrics, an ideal change detection algorithm would produce results with such characteristics: E_c=100%, E_n=0% and ΔN=0 blob. The metrics results are displayed in the next section for the three different sequences presented above.

8.6.3 Results

The foreground objects of the 'dirc', 'lab' and 'ptzets' sequences segmented by the moving object detector, with and without noise filtering (see section 8.4), are evaluated in the following experiments. The system accuracy is evaluated via the E_c metric, the noise level is evaluated via the E_n metric and the number of noisy blobs is calculated via metric ΔN all defined in the previous section. The results of these three metrics are displayed in table 8.1, 8.2 and 8.3 for the moving objects, moving edges and the filtered moving object detectors respectively.

Table 8.1 shows that the foreground objects are well segmented with an accuracy of E_c = 90% for the two sequences captured with the same camera and in the same environment *i.e.* the 'dirc' and 'lab' sequences. The results obtained with the 'ptzets' sequence give a E_c rate of about only 76%. The noise results given show that important levels of noise occur in the 'lab' sequence with $E_n \approx 5\%$ of the frame areas covered by noisy blobs and with a corresponding of $\Delta N \approx 8$ noisy blobs. ΔN remains below one noisy blob per 'dirc' frame with 2.41% of their surfaces covered by noise and approximately 2 noisy blobs occur in the 'ptzets' frames covering 1.27% of the frame surfaces.

The same table is constructed with exclusively the foreground blobs segmented by the moving edge detector in table 8.2. Comparing this table with the previous one shows that the moving edge detector is not as accurate as the moving object detector in terms of object area coverage (E_c). It can be observed that the lowest E_c results are given with the 'ptzets' sequence. Table 8.2 also shows that similar noise level of E_n is obtained for the three sequences and lower than the E_n results in table 8.1. However, the number of noisy edges blobs are in general higher than the number of noisy objects blobs, except for the 'lab' sequence. Noisy blobs segmented by

	'dirc'	'lab'	'ptzets'
E_c (%)	90.01	89.99	76.44
E_n (%)	2.41	5.05	1.27
$\Delta N(blobs)$	0.97	8.26	1.95

Table 8.1: Results of the moving object detection

	'dirc'	'lab'	'ptzets'
E_c (%)	80.45	87.12	62.25
E_n (%)	0.59	0.45	0.54
ΔN (blobs)	5.07	5.19	4.85

Table 8.2: Results of the moving edge detection

	'dirc'	'lab'	'ptzets'
E_c (%)	90.79	90.12	73.64
E_n (%)	0.96	0.74	0.82
ΔN (blobs)	0.26	0.55	0.65

Table 8.3: Results of the filtered moving object detection

the moving edge detector are generally thinner than the moving object detector due to the fact that edges in dark regions are partially segmented creating thin, elongated blobs compared to objects foreground patches. This also explains why there are generally more noisy blobs when edges are segmented.

The results of the filtering process of the moving objects with the moving edges in table 8.3 show that the accuracy obtained in the first table is conserved with lower noise level and lower noisy blobs. The high accuracies obtained with the 'dirc' and 'lab' sequences are slightly improved and remains around 90% whereas the 76.4% accuracy obtained with the 'ptzets' sequence drops to 74.6%. The noise level is decreased to below 1% of the frame surfaces for the three sequences with less than 1 noisy blob. Therefore, although the accuracies are not significantly improved, the noise is greatly decreased when moving blobs are filtered with the segmented moving edges.

The moving edge detector segments mainly the edges and hence objects are not as fully segmented as with the moving object detector (as shown in figure 8.6). Moreover, the moving edge detector segments objects into several small edge objects making the average number of noisy blobs ΔN much greater than with the moving object detector. The only advantage the moving edge detector has is that it does not produce noisy foreground pixels in the sensitive dark regions, as reflected by the value of E_n.

Table 8.3 show that the moving objects results are improved when combined with the moving

edges information in the filtering process described in section 8.4.3. The E_c keeps approximately the same accuracy after filtering while the noise is greatly reduced in terms of covered areas E_n and in terms of number of blobs ΔN.

8.6.4 Discussion

The experimental results of the moving object detection algorithm described in section 8.4 show that 90% of the moving object areas are successfully detected for the two sequences captured in an indoor environment. Only about 74% of several foreground objects areas of an outdoor scene captured by a surveillance camera are segmented. However, although filtering is performed on the dimension of the segmented blobs, noisy measurements are observed with a maximum of 5% of the total frame area and about 8 noisy small blobs appearing essentially at dark regions of the images.

In order to filter out the noisy detection, moving edge are segmented and although their detection is also noisy, the combination of moving objects and edges allow a better reduction of these noisy moving objects. The results show that less than 1% of the total frame area is detected as noise with less than one noisy blob occurring per frame for the three used sequences. The gradient-based global motion estimator described in section 8.3 hence allows a frame-rate and accurate image registration and a foreground object detection but with significant noise that could only be reduced by filtering out small objects and by using moving edges information.

8.7 Conclusion

The robust and global motion estimator developed in chapter 6 and reviewed in section 8.3 can reliably estimate the motion of a background scene modelled by a single planar motion model when small portions of the background are occluded by moving foreground objects. Using motion compensation and a temporal averaging technique to construct a background reference frame, moving objects are successfully segmented from this background and noise is then reduced by using a moving edge detection algorithm.

The segmentation results show that the blobs bounding boxes of the moving foreground objects are relatively accurately segmented with percentages of area coverage between 73 and 90% and which should be evaluated further with additional different sequences. However, the segmentation results are considerably noisy at the darker regions in the images but are greatly improved if small objects are eliminated and if they are combined with the blobs of the moving edges. The small objects filtering and the filtering by the moving edges showed that less than 1% of the whole frame area is covered by noisy blobs and an average less than one small noisy blob occurs per frame.

Chapter 9

Final discussion

The general aim of this thesis is to analyse motion via optical flow techniques and is centred around three main themes. The first theme addresses the problem of contamination caused by noise and the existence of multiple motions occurring in the estimation neighbourhoods. The noise process is caused by electrical and optical imperfections of cameras and multiple motions - typically caused by occlusion. The second theme of this study addresses the estimation of large motions. Gradient-based optical flow techniques are derived from a Taylor series expansion of the constant brightness equation around the small motion assumption. Therefore, they can only estimate accurately pixel motions of small magnitude.

Unless information about the content of a frame is given, motion analysis between frames is best performed if the motion is available at each pixel. Such dense motion information can be computed relatively accurately but is computationally expensive. The third theme of this thesis addresses this time limitation problem by investigating the optical flow technique in the application of global motion estimation in order to detect foreground objects occluding background scenes captured in indoor and outdoor environments. These three themes are investigated in this study and the obtained results are summarised below. This chapter concludes with a discussion on future work which analyses the important work that would render the investigation more complete and that would improve the results for directed future research.

9.1 Summary of the research

A summary of the results obtained throughout this study is outlines in the six following subsections. The first four sections analyse the gradient-based optical flow estimator developed in chapter 3 which provide dense pixel motion estimation. The estimator is analysed from a mathematical point of view, its limitations are addressed and a confidence measurement is provided in the fourth section. The fifth section analyses two on-line global motion estimators: the hyperplane and the gradient-based estimators, while the sixth analyses the results of the estimated motions induced by static background scenes, captured by moving pan and tilt cameras. The

global motion of a zooming camera is also reviewed in this section. Finally, we describe how foreground moving objects can be detected from static background scenes viewed by stationary cameras and moving pan and tilt cameras.

Dense gradient-based optical flow estimation

Optical flow estimation relies on the constant brightness equation (CBE) which states that brightness at any location in space does not change over small time intervals. An optical flow estimator based on this constraint is developed in section 3.3 around the small motion magnitude assumption. Because of the aperture problem, additional constraints are necessary and are provided by fitting a planar motion model to the motion field occurring across local neighbourhoods of pixels. A least-squares regression technique is then applied to minimise the error function associated with the CBE.

Non-iterative and iterative motion estimators are implemented. The iterative estimation approach is shown to be more accurate than the non-iterative approach since it converges to better motion estimates. In section 3.6, a study of the gradient term of the optical flow equation shows that it can be approximated by the widely used and simplest grey-level derivative. Although this approximation improves greatly the speed of the algorithm, the optical flow estimator remains a very expensive technique in providing dense optical flow fields. The estimator remains limited to small motion magnitudes. Also multiple motions within the same neighbourhood - typically occurring at object boundaries - cannot be modelled by a single planar motion model.

Robust statistical approach

Noisy images and multiple motions within neighbourhoods of pixels are likely to occur in real captured datasets. Real-life image sequences contain various sources of noise such as change of illumination, non-uniform reflective properties of 3D surfaces, transparencies and high-frequency noise from the optical system. Such noise disturbs the least-squares minimisation process of the optical flow estimator since the constant brightness assumption is no longer respected. Multiple motions, occurring essentially at object boundaries, do not belong to the same motion field and therefore cannot be all modelled by the same planar motion model and results in motion contamination.

A robust statistical approach was introduced to address these contamination problems. It was shown that such techniques can deal successfully with high levels of noise. Although the motion estimation is improved at object boundaries, the multiple motion problem could not be entirely eliminated and smoothed fields were estimated across these motion boundaries.

Hierarchical framework: multi-resolution estimation

Gradient-based optical flow estimators have a limited range of operations in terms of motion magnitude unless an initialisation technique provides initial estimates of the motions. Such

initialisation techniques are the feed-forward, block-matching and hierarchical techniques. A multi-resolution technique, also referred to as a coarse-to-fine technique, is used in this study to recover large motions. This hierarchical technique constructs a *Laplacian pyramid* of sub-sampled frames. Sub-sampling addresses the problem of large motions since any pixel motion in a frame also gets progressively sub-sampled up the pyramid. Estimation at the top level of the pyramid is initialised with zero motion estimates and requires a large number of iterations whereas each of the lower levels in the pyramid are initialised with the magnified motion field from the corresponding upper level and a few iterations are sufficient to estimate larger motions.

Optical flow uncertainty

Dense fields of motion vectors are of great interest in many computer vision applications but often lack accuracy. A method for providing a confidence measurement was developed in chapter 4 which provides covariance ellipses and confidence. High confidence was obtained at low-textured pixels and at textured areas undergoing a single dominant motion field whereas low confidence was obtained across object boundaries where multiple motions occur and where the constant brightness constancy does not hold.

Investigating the hyperplane approach to calculating optical flow

A recent innovation to the estimation of motion is the hyperplane estimator [49]. A comparison with the gradient-based optical flow estimator was carried out. The hyperplane estimator, generally used for template matching applications, was implemented in chapter 5. The approach learns the relationship between applied motion fields and the respective grey-level differences, also called DFDs (displaced frame differences). Solving for one specific motion is equivalent to solving a set of linear equations in a multi-dimensional space referred to as the hyperplane equations.

The Hyperplane estimator is non-iterative and is compared in chapter 5 to the non-iterative gradient-based estimator adapted for global motion estimation. The results showed that both estimators can accurately estimate motion with small magnitudes. The hyperplane accuracy is highly dependent on a computationally expensive learning process and it can only estimate a limited range of motion magnitudes departing from a reference location. Therefore the estimator's usefulness is limited in terms of camera use. Because of this restriction, the gradient-based estimator is preferred as a global motion estimator since it can estimate motion between pair of successive frames and the small motion magnitude limitation can be addressed by the use of the multi-resolution approach implemented in chapter 4. More importantly, it can be implemented using the robust statistics, iterative and multi-resolution approaches explored in chapter 3 and 4.

Global motion estimation for foreground object detection

Global motion is commonly used to describe the motion of a background scene relative to a camera. The motion due to object movement is referred to as *local motion.* Global motion is used in various applications such as in video coding (MPEG-1/2) or video annotations.

As the hyperplane global motion estimator implemented in chapter 5 was shown to be unreliable, a robust and hierarchical gradient-based version of the estimator developed in chapter 4 was chosen for global motion estimation. This gradient-based technique is performed by fitting a single planar motion model to samples from the entire viewed background scene. This technique can be performed quickly when a small number of edge-based pixels are chosen.

This global motion estimator was evaluated in chapter 6 for two different applications. First, the motion of background scenes captured by moving pan and tilt cameras were estimated and great accuracy was obtained with less than 1 pixel error. However such accuracy is not sufficient for accurate image registrations such as background scene mosaicking which can be obtained with dense optical flow techniques as the one developed in chapter 3 or features tracker such as a block-matcher tracking corner points.

The gradient-based estimator is then used in a zooming application where a zoom motion model is fitted to a static background scene viewed by a fixed camera performing zooming operations. The optical flow estimation allows for the calculation of the centre of expansion *i.e.* the intersection between the optical axis and the image centre. The results show that the location of the centre of expansion varies as the focal length changes during zooming operations.

Tracking with surveillance cameras

The problems associated with tracking remain unsolved since there are many occurring sources of ambiguities caused for example by shadows, illumination changes, over-segmentation of moving objects and object mis-detection. In addition, the high variability often present in the projected images of an object over time makes its tracking difficult. This variability arises from three principle sources: variation in the objects pose, variation in illumination, and partial or full occlusion of the target. When ignored, any of these sources of variability are enough to cause a tracking algorithm to lose its target or to mistrack with others.

Tracking is generally applied for surveillance applications where stationary cameras capture static background scenes as applied in chapter 7. Background models of the static background scene are constructed over time which allows the detection of foreground pixels and subsequently foreground objects. However, when several of these objects occluded each other, their foreground pixels cannot be separated into individual objects and tracking them becomes problematic.

An appearance-model based approach consisting of modelling regions of objects was explored in chapter 7 in order to recover and track the non-occluded grey-level information during occlusion. Although this technique improved the traditional trajectory-model Kalman tracking technique which blindly predicts object position during occlusion, tracking still is not success-

ful. Additional information of the objects, such as knowledge of their 3D positions or colour and shape modelling, is needed for coherent object tracking during occlusion.

Detecting objects in pan and tilt applications

Detecting foreground pixels when cameras are moving (pan and tilt cameras) cannot be performed as reliably as when cameras are stationary, as demonstrated in chapter 8. Motion of background scenes relative to a camera can be well detected by the global motion detector developed in chapter 6 but it is not accurate enough to allow an accurate image registration and to construct accurate models of the background as performed with stationary cameras.

Moving foreground objects are detected using a thresholding operation between captured frames and a motion compensated reference frame, updated using an average process. Moving objects were relatively well segmented but with high associated noise represented by small noisy clusters of foreground pixels across the frame referred to as *blobs*. These noisy blobs were first eliminated by a simple threshold on their dimensions. A second filtering was performed on the noisy blobs by using an overlapping procedure between the blobs corresponding to the moving objects and the blobs detected from a moving edge segmentation technique.

9.2 Contributions

The following sections describe the work contributed by the motion analysis performed in each chapter of the investigation.

Ground-truth generations and evaluation procedures

Evaluating optical flow has never been easy since accurate and dense ground-truth optical flow is rarely available. Producing such information requires a precise knowledge of the 3D scene geometry as well as the internal parameters of the camera capturing this scene. Because this information was not available, synthetical ground-truth motion vectors were created. Additionally, the ground-truth of the translational component of the global motion induced by pan and tilt cameras capturing a background scene was estimated by an expensive block-matching technique - chapter 5 and 6.

The frames displaced by ground-truth vectors were constructed using the bilinear interpolation technique developed in Appendix A. In this Appendix, the bilinear technique was compared to the nearest-neighbour, the quadratic and cubic techniques and was shown to be the best interpolation technique to use for motion compensating frames.

The percentage metric of accurately estimated flows was developed to provide a measurement of the motion estimator's performance. Convergence in motion magnitude errors was also defined to evaluate the final accuracy of measurements and speed of estimators.

True optical flow gradient term and approximations

All optical flow estimators in the literature use several approximations to formulate motion in the image plane including the gradient term. The true expression of this gradient term was derived in chapter 3 showing the complexity of its formulation. It was implemented and discussed. Five approximations were created including the major ones used in the literature and were compared in an evaluation process. The conclusions showed that the widely used approximation of the gradient term in the literature is the quickest and whose accuracy is as good as the mathematically rigorous expression, as far as the motion magnitude to be estimated is within a few pixels.

Confidence expression of optical flow in a multi-resolution and robust framework

A novel expression of the covariance of the optical flow expression was derived in chapter 4 which allow to provide the user with confidence measurement of the estimated optical flow. This covariance could be displayed in terms of areas of confidence by ellipses of uncertainty. And confidence measurements could prove the gain of accuracy when the multi-resolution and robust framework is used. The confidence measurements were geometrically interpreted through the use of ellipses, or rectangles framing these ellipses, which represent the domain of uncertainty around optical flow vectors.

Edge-based sampling technique for accurate global motion estimation

Estimating motion of background scenes captured by moving pan an tilt cameras using the gradient-based estimator is very computationally expensive if the entire set of pixels representing the background scene is chosen as the neighbourhood of constraining pixels. The number of pixels was instead restrained to be located onto a spare set of edges belonging to the background scene.

A novel technique was then developed to automatically sample pixels from the background pixels with the highest gradient information (strongest edges). An expensive edge-based sampling technique was designed in chapter 5 to evaluate the hyperplane and gradient-based global motion estimator. A quicker and similar edge-based technique was then designed in chapter 8 for the gradient-based moving object detector.

Determination of the centre of expansion of a zooming camera

The gradient-based global motion estimator was also used to estimate the zooming motion parameters in a zooming application which could allow the determination of the varying positions of the centre of expansion (intersection of the optical axis with the image plane). The estimated

centred of expansions were shown to vary with the focal length of a fixed camera when zooming operations are performed.

Segmentation of moving object in pan and tilt applications

The gradient-based global motion estimator was used in chapter 8 to segment foreground moving objects occluding background scenes captured by moving pan and tilt cameras. The amount of foreground pixels is small enough so that the robust statistical approach developed in chapter 4 could reject them from the optical flow estimation. The results were shown to be relatively accurate and moving objects were successfully segmented (but with considerable noise). A simple local background reference frame was built over time using a grey-level averaging approach and foreground edge-based pixels were segmented to filter out the noisy detection. The accuracy remained unchanged after the filtering process while noise was greatly reduced. This strategic on-line foreground segmentation can allow for example future tracking possibilities.

Hybrid object tracking with stationary cameras

Traditional trajectory-based trackers are shown to often loose tracks of objects when they interact with each other when their foreground pixels cannot be separated easily into individual clusters of foreground pixels *i.e.* blobs. An appearance-model tracker was then designed to track segmented grey-level regions of objects during occlusion. This tracker was implemented in a hybrid scheme with a trajectory-based tracker for better tracking results but were shown insufficient for successful tracking.

9.3 Future work

Future work can be grouped into three main components. First the accuracy of the dense optical flow estimator can be improved as well as its speed. Second, global motion of moving pan and tilt cameras is accurate enough for foreground object detection but inefficient for accurate background frame registration. This limitation can be addressed if a motion segmentation technique is achieved. Finally, tracking objects in static background scenes during occlusions could be performed more reliably, if, for example, shape and colour models are used and 3D location are known. Such 3D positions could be obtainable if, for example, stereo systems are used. These ideas are discussed in more detail below.

Use of confidence for accuracy and speed increase

The presence of multiple motions within a local neighbourhood of pixels is the major problem that causes optical flow algorithms to fail to estimate motion across object boundaries accurately. Robust statistics was introduced in chapter 4 to address this problem but could not eliminate successfully outlying motions from contaminating the estimation of the dominant motion field.

Since gradient-based algorithms are built upon the constancy of brightness, obtaining a lack of accuracy in the motion estimates does not automatically mean that the constant brightness constancy is violated on a single pixel basis. However, when such grey-level errors are considered for an entire neighbourhood of pixels, lack of accuracy is easily detected by the confidence measurement developed in section 4.3. Combining confidence with the information about edge location could allow a refinement of pixel motion results.

The dense optical flow estimator was developed to provide a motion vector to each pixel in an image using the pixel neighbourhood. This means that neighbourhoods overlap by a large amount in the motion estimation from one pixel to a neighbouring pixel. A planar motion model is fitted to each pixel, therefore all pixel motions within a neighbourhood are known. A semi-dense, and hence a faster, scheme could be then implemented by avoiding the re-calculation of the motion of pixels present in neighbourhoods when high confidence is obtained. More dense estimation would be required at low confidence regions such as across object boundaries where multiple motions occur. In addition, the neighbourhood size and its shape could be adjusted in such a way that it contains a minimum required number of pixels and that its shape is adjusted according to the shape of motion boundaries. For instance, the fixed square shape could be replaced by a varying elliptical shape so its circumference fits to the local object boundary and motion contamination is thus minimum.

Computational speed is easily improved with special hardware design. For example, a DSP card was designed for the PRISMATICA [1] project which allowed real-time computation of sparse optical flow vectors at every 8×8 pixels on sub-sampled images. Using the semi-dense technique previously described, optical flow could be obtained at approximately every maximum 10×10 pixels according to the results of chapter 3.

Most typical optical flow techniques presume brightness constancy, which is often violated by time-dependent physical processes. For example, grey-level changes of non-moving pixels can be interpreted as pixel motion when the orientation of the illuminant changes with respect to the object surface orientation or when the illuminant radiations are altered by physical processes. Additional model of image brightness variation are often used with success [64, 141, 63] and could be implemented in our algorithm to address this problem.

Global motion accuracy

Using a sparse set of edge-based pixels as pixels neighbourhood, global motion of moving pan and tilt cameras could be accurately estimated by the gradient-based technique described in chapter 6. However, although the results allowed consistent foreground object detection, the estimator's accuracy was not high enough to construct an accurate background as achieved with

[1] http://www.prismatica.com/

stationary camera applications. This is mainly due to the fact that a single planar motion model was fitted to the background scene assumed to be modelled as a single 3D surface. However, backgrounds generally consist of multiple objects and better global motion estimation could be obtained if these objects were segmented into individual 3D surfaces. An iterative process could be then implemented to gain higher accuracy and mosaics of the background could be constructed.

Tracking occluded objects

The tracking algorithm developed in chapter 7 could not track successfully objects in occlusion scenarios. A shadow detection and removal algorithm would improve the results [39, 75] but additional information about the objects is required for successful tracking. For example, the grey-level segmented regions in chapter 7 could be segmented in a colour space and their shape could be modelled. More importantly, the 3D location of the tracked objects would improve the results greatly. The optical flow developed in this study can be used to calculate the disparity of certain feature points as shown in the frame example[2] of figure 9.1. The disparity measurements are calculated at some Harris corner pixels [62]. The corner strength of a pixel is represented by a square area centred on that pixel in the figure where large areas mean strong corner responses. This stereo information could also be used to process 3D information of static and moving objects where Harris corners strength can be replaced by the optical flow confidence measurements.

[2]http://vasc.ri.cmu.edu//idb/html/motion/hotel/index.html

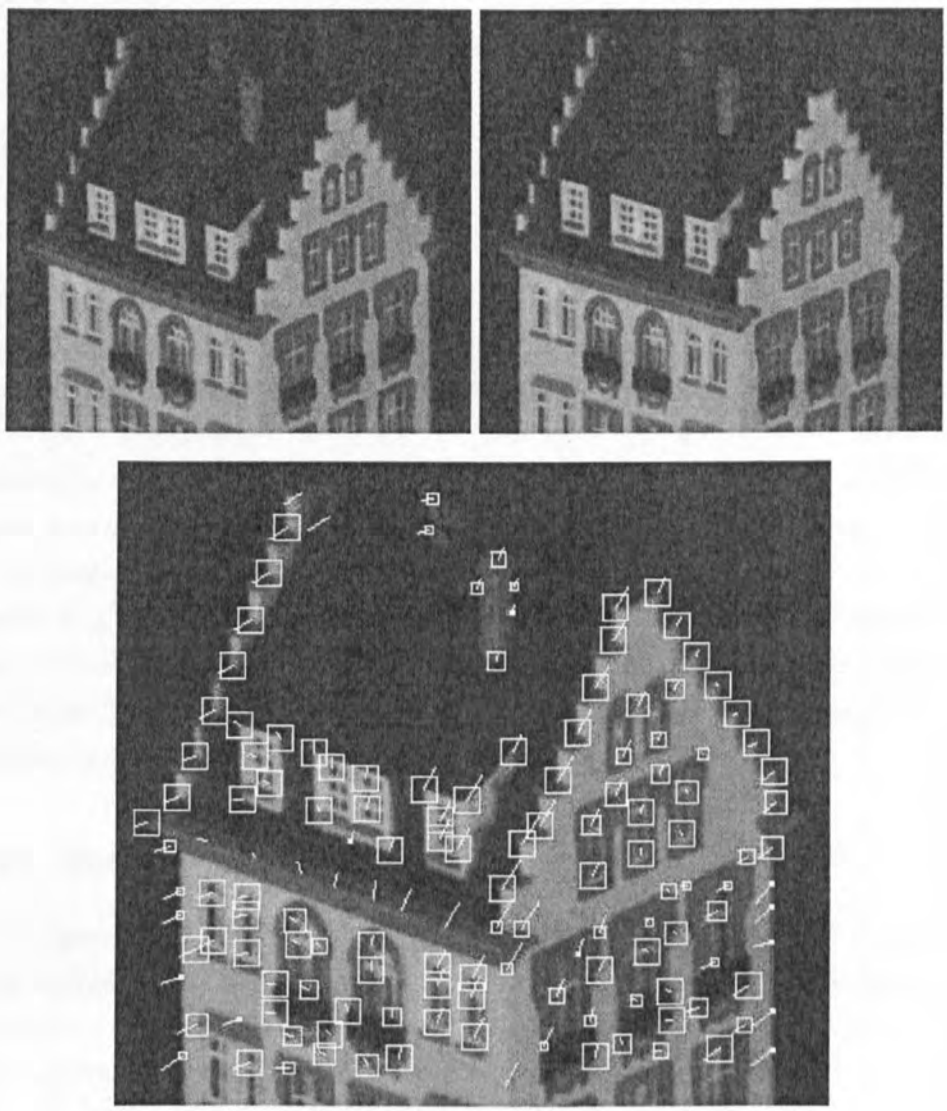

Figure 9.1: Top: stereo pair with bottom: spare disparity measurements with associated confidence

Appendix A

Interpolation

In many areas ranging from cartography, digital imaging and modelling, and engineering applications, data collected from the field are usually discrete. Interpolation techniques aim to reconstruct data using local information for re-sampling purposes such as resizing an image to desired scale. The process of estimating the values in between sampled data points is called *interpolation*; whereas the process of estimating the outcomes beyond the range covered by the existing data is called *extrapolation*. Optical flow computation often requires motion compensating frames to compare the grey-levels of two frames displaced by estimated optical flow. The accuracy of optical flow algorithms is directly dependent on the accuracy of the reconstructed frame. Four interpolating techniques are compared in this appendix in motion compensation experiments. These techniques are known as the *nearest-neighbour*, *bilinear*, *quadratic* and *cubic* techniques and are described in the four following sections.

A.1 Nearest-neighbour interpolation

The simplest interpolation scheme is the so-called zero-order or nearest-neighbour interpolation. In this method, the grey-level of an output pixel is taken to be that of the input pixel nearest the location to which the output pixel maps. Using the image grid notation in figure A.1, the nearest-neighbour interpolated grey-level $I(\mathbf{x})$, at location $\mathbf{x}$, is expressed as

$$I(\mathbf{x}) \approx I(\mathbf{x}') \tag{A.1}$$

where

$$\mathbf{x}' = \text{argmin}_{\mathbf{x}' \in N(\mathbf{x})} \|\mathbf{x}' - \mathbf{x}\| \tag{A.2}$$

$$N(\mathbf{x}) = \{\mathbf{o}(\mathbf{x}), \mathbf{d}(\mathbf{x}), \mathbf{v}(\mathbf{x}), \mathbf{h}(\mathbf{x})\} \tag{A.3}$$

and

$$\mathbf{o}(\mathbf{x}) = [floor(x), floor(y)]^T$$

$$\begin{aligned} \mathbf{v}(\mathbf{x}) &= [ceil(x), floor(y)]^T \\ \mathbf{h}(\mathbf{x}) &= [floor(x), ceil(y)]^T \\ \mathbf{d}(\mathbf{x}) &= [ceil(x), ceil(y)]^T \\ floor(x) &= \text{greater integer not greater than} \\ ceil(x) &= floor(x) + 1 \end{aligned} \tag{A.4}$$

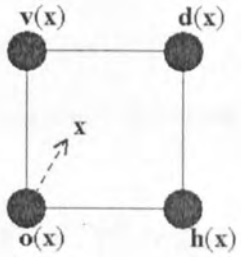

Figure A.1: Neighbourhood of the nearest-neighbour and bilinear interpolators

The nearest-neighbour interpolation technique is computationally simple and can produce acceptable results. However, it introduces artefacts in images containing fine structures where the grey-level is likely to change significantly from one pixel to the next.

A.2 Bilinear interpolation

Bilinear interpolation produces more desirable results than the previously described zero-order interpolation technique with a relatively small increase in programming complexity and execution time. The bilinear function $f(\mathbf{x})$ is defined such that a non-planar grey-level surface passes through the four neighbouring points $\mathbf{o}(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$, $\mathbf{v}(\mathbf{x})$ and $\mathbf{d}(\mathbf{x})$ defined in equations A.4 and interpreted in figure A.1. The surface $f(x,y)$ is fitted to the four points by the following linear equation

$$f(x,y) = ax + by + cxy + d \tag{A.5}$$

Any grey-level located at $\mathbf{x}$ between four the neighbouring pixels can be interpolated alternatively by calculating

$$\begin{aligned} I(\mathbf{x}) &= (1-\Delta x)\,(1-\Delta y)\,I\,(\mathbf{o}(\mathbf{x})) \\ &+ (1-\Delta x)\,\Delta y\,I\,(\mathbf{h}(\mathbf{x})) \\ &+ \Delta x\,(1-\Delta y)\,I\,(\mathbf{v}(\mathbf{x})) \\ &+ \Delta x\,\Delta y\,I\,(\mathbf{d}(\mathbf{x})) \end{aligned} \tag{A.6}$$

where

$$\Delta\mathbf{x} = [\Delta x, \Delta y]^T = \mathbf{x} - \mathbf{o}(\mathbf{x}) \tag{A.7}$$

A.3 Higher-order techniques: quadratic and cubic interpolation

The smoothing effect of bilinear grey-level interpolation may degrade fine details in the image. Extra computational efforts of higher order interpolation techniques, such as the quadratic and cubic interpolator, better preserve fine details by enlarging the interpolating grid. The quadratic and cubic fitting of a function $f(\mathbf{x})$ are described by

$$f(\mathbf{x}) = X(\mathbf{x})\ \mathbf{a} \tag{A.8}$$

where $\mathbf{a}$ is the unknown vector of unknown parameters to be estimated. For the quadratic interpolator

$$\begin{aligned} X(\mathbf{x}) &= [x^2, xy, y^2, x, y, 1] \\ \mathbf{a}^T &= [a_0 ... a_5] \end{aligned} \tag{A.9}$$

and for the cubic interpolator

$$\begin{aligned} X(\mathbf{x}) &= [x^3, x^2y, y^2x, y^3, x^2, xy, y^2, x, y, 1] \\ \mathbf{a}^T &= [a_0 ... a_9] \end{aligned} \tag{A.10}$$

The interpolation vector $\mathbf{a}$ contains $n = 6$ parameters for the quadratic method and $n = 10$ for the cubic method, therefore a minimum number of 6 and 10 input pixels (respectively) are necessary for $\mathbf{a}$ to be estimated. The n equations can be re-written in terms of matrices as

$$F = P\mathbf{a} \tag{A.11}$$

where

$$F = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix}, \text{ and } P = \begin{bmatrix} X(\mathbf{x}_1) \\ \vdots \\ X(\mathbf{x}_n) \end{bmatrix} \tag{A.12}$$

From equation A.11, the interpolating vector can be estimated by $\mathbf{a} = P^{-1}F$ for the quadartic case and by

$$\mathbf{a} = \left(P^T P\right)^{-1} P^T F \tag{A.13}$$

for the cubic case as the P matrix is not a square matrix in this latter case. The 6 pixels required for the estimation of the quadratic vector $\mathbf{a}$ are situated around $\mathbf{x}$ according to the left figure of A.2. In this figure, the central pixel $\mathbf{x}_3$ is first located as the closest pixel to $\mathbf{x}$, $\mathbf{x}_{1,2,4 and 5}$ are then located as a cross around $\mathbf{x}_3$ and the last pixel $\mathbf{x}_6$ is the closest pixel to $\mathbf{x}$ non-overlapping with any other neighbours. The cubic interpolation technique is an extension of the quadratic technique with cubic terms. The number of points required to solve the set of equations is ten and the system is chosen to be over-constrained by adding two extra input pixels. The twelve pixel locations are displayed so they represent a cross around $\mathbf{x}$ as displayed in figure A.2.

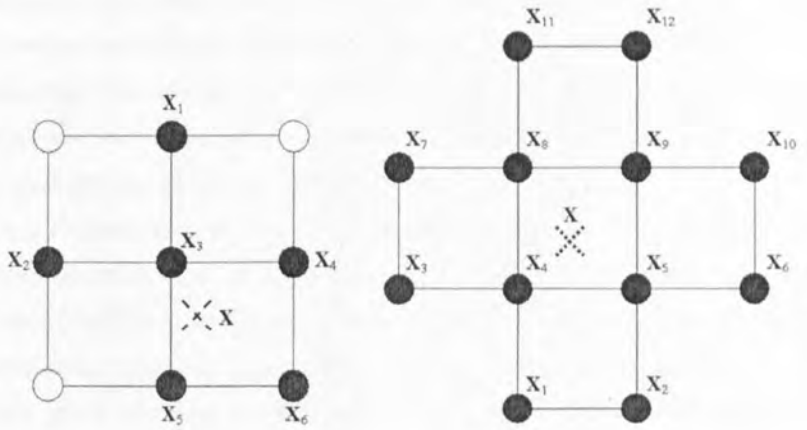

Figure A.2: Neighbourhood of the left: quadratic and right: cubic interpolator

A.4 Interpolation experiments

The accuracy of the four interpolating techniques described in the previous sections are evaluated in the following motion compensation experiments. Two-dimensional sine grey-level patterns are created from a horizontal and vertical sine signal with different periods. These sine frames are then displaced by a known arbitrary motion vector and reconstructed by the interpolating techniques. The mean grey-level error between the frames and the reconstructed frames are recorded and plotted in figure A.4.

The periods of both horizontal and vertical sine signal are chosen equal. Frame examples are shown in figures A.3 with a period of 5, 10, 15, 20, 25, 30, 35 and 40 pixels from the top to bottom and left to right figures respectively. Signal with a very large period means that the frame contains few textures, low gradient contents whereas smaller period frames contain higher gradients pixels.

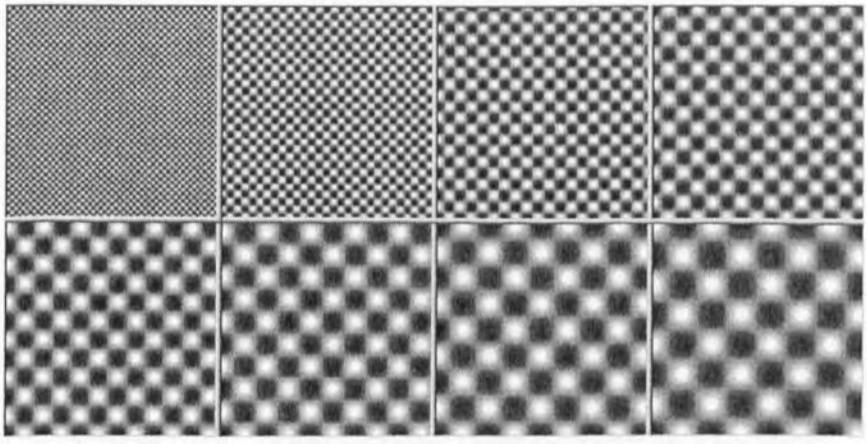

Figure A.3: Sine frames

The motion compensation results in figure A.4 show that the nearest-neighbour technique does not converge to a minimum error value for periods less than 40 pixels. The mean grey-level errors remain high with a value of 2 for 40 pixels period frames. The three other techniques, the bilinear, quadratic and cubic techniques all reconstructs accurately frames with errors less than half a grey-level unit for all frames with periods greater or equal than 15 pixels for the bilinear technique and 10 pixels for the quadratic and cubic techniques. The two latter techniques are the most accurate techniques for all frequencies frames. However, while the cubic technique is slightly more performant than the quadratic method the quadratic interpolator is better for frames having small periods less than 10 pixels approximately. This difference of accuracy for high gradients pixels is mainly due to the fact that the size of the neighbourhood of the cubic interpolator (maximum 4 pixels across, see right figure A.2) can be too large to reconstruct some parts of the sine signals and the cubic model cannot be fitted accurately to the signal. Hence, reconstruction errors result for small period signals as the interpolation model gets more sophisticated.

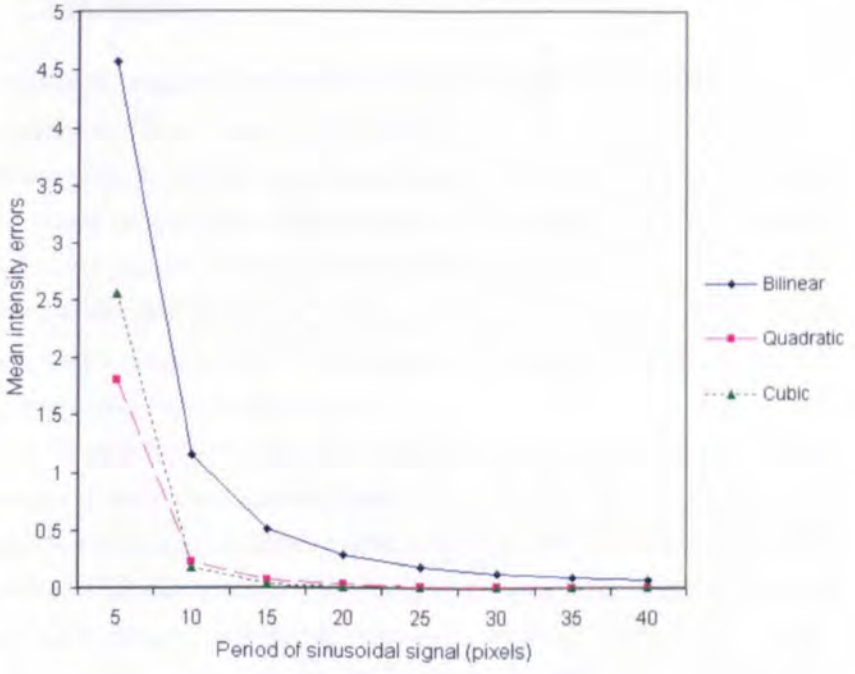

Figure A.4: Mean Error of four interpolating techniques

To conclude the experiments, the computational time required to reconstruct a frame of 600×600 pixels, so 360,000 pixels, by the four different interpolation is measured and displayed in table A.1. It shall be noted that the inverse matrix operation involved in the quadratic and cubic interpolation requires to be performed only once as matrix P remains the same at all times (see equation of matrix P in A.12). Hence, the time required to perform P^{-1} for the quadratic case and $(P^TP)^{-1}P^T$ is omitted in the data of table A.4. The results of this

figure show how expensive the quadratic and cubic techniques are when motion compensating a frame of standard size. The simplest technique, the nearest-neighbour method is the quickest technique but lacks accuracy. Therefore, the bilinear technique which has a similar complexity as the nearest-neighbour performs as accurately as higher order techniques is the most appropriate technique to use for reconstructing pixel patterns in computer vision. Despite the small lack of accuracy for gradients of small periods, it is an acceptable technique for most kind of frames content.

	Interpolation technique			
	cubic	quadratic	bilinear	nearest-neighbour
time in seconds	11.27	5.3	0.85	0.81

Table A.1: Expenses of different interpolation techniques for a 360,000 pixels frame

A.5 Conclusion

Four interpolating technique are implemented in this annex and compared in a motion compensation experiments. These interpolation techniques are from the lowest to highest order model the nearest-neighbour, bilinear, quadratic and cubic techniques. The nearest-neighbour interpolates by taking the grey-level of the closest pixel, the bilinear fits a non-planar surface into a neighbourhood of 4 pixels, the quadratic and cubic technique involve quadratic and cubic terms in the fitting of the grey-level pattern of 6 and 12 pixels respectively. Experiments are run for synthetical frames constructed from sine signals with varying frequencies to simulate varying gradients information of real images.

The results show that the nearest-neighbour is the quickest but is too inaccurate in the reconstruction of motion compensated frames. The most accurate technique, the cubic interpolator, requires a significant number of input points for the estimation to be possible and this causes erroneous reconstructionof local grey-level patterns in high frequency textured images. The quadratic technique interpolation technique has similar high accuracy as the cubic technique and does not have the same restriction in reconstructing high gradients pixels. However, the complexity of this technique rendres this technique non-appropriate for real-time motion compensation application. The remaining technique, the bilinear interpolation, is the quickest technique after the nearest-neighbour and performs in real-time and has the best performance for all frequencies occurring in image data as the cubic technique with a slight drop of accuracy in high gradients pixels. Therefore the bilinear interpolation technique is the most suitable technique for motion compensation applications.

Appendix B

Kalman filtering for trajectory prediction

Tracking objects is often performed by the use of the Kalman filtering technique. A Kalman filter can predicts objects positions according to covariance matrices built from the measurements of errors between predicted and observed positions [25, 121, 139]. Section B.1 describes the two steps involved in the Kalman filtering process: the prediction and the update step. The second section B.2 implements a simple Kalman filtering for tracking 2D objects based only on position measurement.

B.1 The underlying equations

Kalman filters are recursive linear minimum mean square filters, meaning that under the assumption that the noise measurement and the noise system are Gaussianly distributed, recursive filters are the optimal estimators in the sense of minimum variance. For a given sequence of observations it is the task of a Kalman filter to estimate the system state at the current time of observation t.

Kalman filtering methods rely on two main equations, the *dynamic equation* relating the state vector p between two successive measurement times: $t-1$ and t, and the *observation equation* relating the measurement vector x to the state vector. The goal is to estimate a process assumed random. Once estimated, the state parameters are updated for the estimation at the next observation time. The random process to be estimated is modelled by the dynamic equation B.1:

$$\mathbf{p}_t = A\mathbf{p}_{t-1} + \mathbf{w}_t \tag{B.1}$$

where A is the state transition matrix and w_t is the white noise associated with the dynamic process. The observations are expressed linearly in terms of the state vector as follows

$$\mathbf{x}_t = H\mathbf{p}_t + \mathbf{z}_t \tag{B.2}$$

where $\mathbf{x}_t$ is the dimensional noisy and distorted observation vector at time t, H is the observation matrix and $\mathbf{z}_t$ is the measurement noise vector. The random variables $\mathbf{w}_t$ and $\mathbf{z}_t$ represent the process and measurement noise respectively. They are assumed to be independent of each other, white and with normal probability distributions with zero mean $N(0,Q)$ and $N(0,R)$. The process noise covariance Q and measurement noise covariance R matrices are calculated by

$$Q = E\{\mathbf{w}_t\mathbf{w}_t^T\} \tag{B.3}$$

$$R = E\{\mathbf{z}_t\mathbf{z}_t^T\} \tag{B.4}$$

In parctice, Q and R change over time, however they are assumed constant.

B.1.1 The prediction step

In the prediction step of a Kalman filter, the predicted state vector $\tilde{\mathbf{p}}_t$ and the predicted state covariance matrix $\tilde{P}_t$ at time t are estimated from the previous at time $t-1$ by

$$\tilde{\mathbf{p}}_t = A\mathbf{p}_{t-1} \tag{B.5}$$

$$\tilde{P}_t = AP_{t-1}A^T + Q \tag{B.6}$$

The Q matrix, defined in equation B.3, is often an boosted covariance matrix to make the Kalman filter a better tracker and more tolerant to the variability in measured object positions.

B.1.2 The update step

The state process $\mathbf{p}_t$ and its covariance P_t are updated from an estimated process $\tilde{\mathbf{p}}_t$ with covariance $\tilde{P}_t$ (equations B.5 and B.6 respectively) according to the Kalman gain K

$$\mathbf{p}_t = \tilde{\mathbf{p}}_t + K(\mathbf{x}_t - H\tilde{\mathbf{p}}_t) \tag{B.7}$$

$$P_t = \tilde{P}_t - KH\tilde{P}_t \tag{B.8}$$

where $\mathbf{p}_t$ is given by equation B.5 and $\tilde{P}_t$ by equation B.6. The Kalman gain is derived by minimising the state error covariance P_t

$$P_t = E\{(\mathbf{p}_t - \tilde{\mathbf{p}}_t)(\mathbf{p}_t - \tilde{\mathbf{p}}_t)^T\} \tag{B.9}$$

and after minimisation

$$K = \tilde{P}_t H^T(H\tilde{P}_t H^T + R)^{-1} \tag{B.10}$$

where R is the measurement error covariance matrix defined in equation B.4. The Kalman gain is defined so it increases with the error between the predicted and actual measurements. Hence as the measurement error covariance increases, the Kalman gain decreases and the state process is principally updated with the predicted state and vice versa.

B.2 Kalman filter implementation for object tracking

The Kalman filter used to track objects in chapter 7 is implemented by a prediction and update algorithm described in the section B.1. The computational time required to perform a Kalman filtering is mainly dependent on the time required to perform the matrix inversion involved in the calculation of the Kalman gain K in equation B.10. Matrix inversion operations are performed in this study by an SVD (Singular Value Decomposition [103]) algorithm which computational expensiveness increases cubicly with the lateral size of the square covariance matrix. However, tracking objects involves relatively expensive operations such as noise removal by a smoothing operation, background reference frame construction involving large memory handlings and many arithmetic operations to detect and segment foreground objects. Thus, in order to limit the overall computational speed of the tracking algorithm, the state vector $\mathbf{x}$ and measurement vector $\mathbf{s}$ are chosen to be of the simplest form where $\mathbf{x}$ contains only the object positions and $\mathbf{p}$ involves only the position and velocity of these objects:

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t \end{bmatrix}^T \tag{B.11}$$

$$\mathbf{p}_t = \begin{bmatrix} x_t & u_t & y_t & v_t \end{bmatrix}^T \tag{B.12}$$

where the object velocity is $\mathbf{v}_t = [u_t, v_t]^T$. A constant velocity trajectory model is chosen which is equivalent to a zero-acceleration model:

$$\tilde{\mathbf{v}}_t = \mathbf{v}_{t-1} \tag{B.13}$$

$$\tilde{\mathbf{x}}_t = \mathbf{x}_{t-1} + \Delta t \tilde{\mathbf{v}}_t \tag{B.14}$$

$$= \mathbf{x}_{t-1} + \Delta t \mathbf{v}_{t-1} \tag{B.15}$$

The expression of the transform matrix A and observation matrix H are obtained from equations B.1 and B.2:

$$\tilde{\mathbf{p}}_t = A\mathbf{p}_{t-1} \tag{B.16}$$

$$\tilde{\mathbf{x}}_t = H\tilde{\mathbf{p}}_t \tag{B.17}$$

According to equation B.11 and B.12, matrix A and H take the following form:

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.18}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.19}$$

Given a known state process at time $t-1$, the prediction and update equations can be calculated if the error covariance matrices Q and R defined in equations B.6 and B.10 respectively are

known. They are here empirically estimated for best tracking as constant matrices as

$$Q \approx \begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix} \tag{B.20}$$

$$R \approx \frac{1}{6} \begin{bmatrix} H & 0 \\ 0 & W \end{bmatrix} \tag{B.21}$$

where W and H are the width and height of the object respectively.

Appendix C

Foreground and background Bayesian classifier

Let's consider a surveillance camera capturing at time t a frame with intensity distribution I_t and a background frame of the non-foreground objects *Back*. It is possible to segment pixels located at $\mathbf{x}$ either as background or foreground according to the Bayesian classifier [60] of equation C.1:

$$\lambda^*(\mathbf{x}) = \text{argmax}_{\lambda(\mathbf{x})\in\{F,B\}} p(\lambda(\mathbf{x})/\Delta I(\mathbf{x})) \tag{C.1}$$

where

$$\Delta I(\mathbf{x}) = |I_t(\mathbf{x}) - Back_t(\mathbf{x})| \tag{C.2}$$

and λ^* is the estimated class the pixel belongs to depending on the grey-level difference with the background: $\Delta I(\mathbf{x})$. In this example $\lambda(\mathbf{x})$ is either the foreground class $\lambda(\mathbf{x}) = F$ or background class $\lambda(\mathbf{x}) = B$. The shadow class which represent the pixels covered by objects shadow is often introduced in many techniques. The term $p(\lambda(\mathbf{x})/\Delta I(\mathbf{x}))$ represents the probability that pixel at $\mathbf{x}$ belongs to the class $\lambda(\mathbf{x})$ given the grey-level difference $\Delta I(\mathbf{x})$. This posterior probability is estimated accoding to the Bayes rule:

$$p(\lambda(\mathbf{x})/\Delta I(\mathbf{x})) = p(\Delta I(\mathbf{x})/\lambda(\mathbf{x}))\frac{P(\lambda(\mathbf{x}))}{P(\Delta I(\mathbf{x}))} \tag{C.3}$$

where $p(\Delta I(\mathbf{x})/\lambda(\mathbf{x}))$ is the conditional probability which can be estimated via the joint probability $p(\Delta I(\mathbf{x}), \lambda(\mathbf{x}))$:

$$p(\Delta I(\mathbf{x})/\lambda(\mathbf{x})) = \frac{p(\Delta I(\mathbf{x}), \lambda(\mathbf{x}))}{P(\lambda(\mathbf{x}))} \tag{C.4}$$

The term $P(\lambda(\mathbf{x}))$ represents the prior knowledge of the class where for example in certain surveillance applications the probability of a pixel to be a foreground pixel can be fixed to be of about 10% and because there are only two classes the probability of this pixel to belong to the background is therefore 90%: see equation C.5 and C.6. These priors can be set dynamically by setting $P(F)$ to be the ratio of the total number of foreground pixels previously measured if

available to the total number of pixels, and $P(B) = 1 - P(F)$.

$$P(F) = 0.1 \tag{C.5}$$

$$P(B) = 0.9 \tag{C.6}$$

The conditional probability of having a grey-level difference given the knowledge that the pixel is a foreground pixel can be estimated in the simplest way by assuming the content of the image to be a random distribution of pixels. Hence the probability density function is approximated to be uniform and every pixel has an equal probability of 1/256 (images are quantised to 256 levels) of having a grey-level difference from the background. This assumption is not true but close to reality and can be approximated by equation C.7. This can be verified by a simple experiment which consists in creating the probability density function *pdf* (a grey-level histogram) of many random captured images and the result would show that real captured images display a pdf rather uniform with less probabilities in the low and high differences (toward the 0 and 256 intensities respectively) than elsewhere where maximum probabilities are obtained around differences of 127 grey-levels.

$$C = p(\Delta I(\mathbf{x})/F) = 1/256 \tag{C.7}$$

If a pixel is known to be background and the background frame is statistically constructed by a Gaussian modelling technique such as the Stauffer and Grimson technique (see description ion section 7.2.1), the conditional probability of having a grey-level difference ΔI with the background is then modelled by a Gaussian distribution of mean $\mu(\mathbf{x}) = Back_t(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$ in equation C.9:

$$p(\Delta I(\mathbf{x})/B) = \frac{1}{\sqrt{2\Pi}\sigma(\mathbf{x})} e^{-\frac{(I_t(\mathbf{x})-\mu(\mathbf{x}))^2}{2\sigma^2(\mathbf{x})}} \tag{C.8}$$

$$= K(\mathbf{x}).e^{-\frac{\Delta I(\mathbf{x})^2}{2\sigma^2(\mathbf{x})}} \tag{C.9}$$

where

$$K(\mathbf{x}) = \frac{1}{\sqrt{2\Pi}\sigma(\mathbf{x})} \tag{C.10}$$

Using equations C.5, C.6, C.7 and C.9 the maximisation of the Bayesian classifier of equation C.1 can be developed as follows

$$\begin{aligned}
\lambda^*(\mathbf{x}) &= \text{argmax}_{\lambda(\mathbf{x})\in\{F,B\}} \left(p(\Delta I(\mathbf{x})/\lambda) \frac{P(\lambda)}{P(\Delta I(\mathbf{x}))} \right) \\
&= \max \left\{ p(\Delta I(\mathbf{x})/F) \frac{P(F)}{P(\Delta I(\mathbf{x}))}, p(\Delta I(\mathbf{x})/B) \frac{P(B)}{P(\Delta I(\mathbf{x}))} \right\} \\
&= \max \{ p(\Delta I(\mathbf{x})/F)P(F), p(\Delta I(\mathbf{x})/B)P(B) \} \\
&= \max \left\{ C.P(F), K(\mathbf{x}).e^{-\frac{\Delta^2 I(\mathbf{x})}{2\sigma^2(\mathbf{x})}} P(B) \right\} \\
&= \max \left\{ log(C.P(F)), log(K(\mathbf{x}).e^{-\frac{\Delta^2 I(\mathbf{x})}{2\sigma^2(\mathbf{x})}} P(B)) \right\} \\
&= \max \left\{ logC + logP(F), logK(\mathbf{x}) - \frac{1}{2} \left(\frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})} \right)^2 + logP(B) \right\}
\end{aligned}$$

$$
\begin{aligned}
&= \max\left\{logC + logP(F) - logK(\mathbf{x}) - logP(B), -\frac{1}{2}\left(\frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})}\right)^2\right\} \\
&= \min\left\{-2(logC + logP(F) - logK(\mathbf{x}) - logP(B)), \left(\frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})}\right)^2\right\} \\
&= \min\left\{2(logK(\mathbf{x}) + logP(B) - logC - logP(F)), \left(\frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})}\right)^2\right\} \\
&= \min\left\{\sqrt{(2)}(logK(\mathbf{x}) + logP(B) - logC - logP(F))^{1/2}, \frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})}\right\} \\
&= \min\left\{\left(2log\left(\frac{K(\mathbf{x}).P(B)}{C.P(F)}\right)\right)^{1/2}, \frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})}\right\} \qquad \text{(C.11)}
\end{aligned}
$$

According to equation C.11, pixel at $\mathbf{x}$ is a foreground pixel if the following equation is true:

$$
\lambda^*(\mathbf{x}) = \begin{cases} F \text{ if } \frac{\Delta I(\mathbf{x})}{\sigma(\mathbf{x})} > \text{Threshold} \\ B \text{ else} \end{cases} \qquad \text{(C.12)}
$$

where

$$
\text{Threshold} = \left(2log\left(\frac{K(\mathbf{x}).P(B)}{C.P(F)}\right)\right)^{1/2} \qquad \text{(C.13)}
$$

Appendix D

3D velocity with small rotation angles

An 3D object point $\mathbf{X} = [X, Y, Z]^T$ moving from time t to time $t + \Delta t$ where Δt is small can be described by a rotation followed by a translation as follows

$$\mathbf{X}(t + \Delta t) = R\mathbf{X}(t) + T \tag{D.1}$$

where $T = [T_X, T_Y, T_Z]^T$ is the translation vector where T_X, T_X and T_Z are the translational components along the X, Y and Z directions. The 3D rotation in equation D.1 is represented by the R matrix expressed by

$$R = R_X R_Y R_Z \tag{D.2}$$

$$R = \begin{bmatrix} c\Theta_Y c\Theta_Z & -c\Theta_Y s\Theta_Z & s\Theta_Y \\ s\Theta_X s\Theta_Y c\Theta_Z + c\Theta_X s\Theta_Z & -s\Theta_X s\Theta_Y s\Theta_Z + c\Theta_X c\Theta_Z & -s\Theta_X c\Theta_Y \\ -c\Theta_X s\Theta_Y c\Theta_Z + s\Theta_X s\Theta_Z & c\Theta_X s\Theta_Y s\Theta_Z + s\Theta_X c\Theta_Z & c\Theta_X c\Theta_Y \end{bmatrix} \tag{D.3}$$

where $c\Theta = cos\Theta$ and $s\Theta = sin\Theta$ and R_X, R_Y and R_Z are the matrices for rotations about the X, Y, and Z axis respectively are expressed below

$$R_X(\Theta_X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\Theta_X & -s\Theta_X \\ 0 & s\Theta_X & c\Theta_X \end{bmatrix} \tag{D.4}$$

$$R_Y(\Theta_Y) = \begin{bmatrix} c\Theta_Y & 0 & s\Theta_Y \\ 0 & 1 & 0 \\ -s\Theta_Y & 0 & c\Theta_Y \end{bmatrix} \tag{D.5}$$

$$R_Z(\Theta_Z) = \begin{bmatrix} c\Theta_Z & -s\Theta_Z & 0 \\ s\Theta_Z & c\Theta_Z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{D.6}$$

For small angles of rotation: $sin\Theta \approx \Theta$ and $cos\Theta \approx 1 - \Theta^2/2 \approx 1$, where *Theta* represents the three angles Θ_X, Θ_Y and Θ_Z, and the 3D rotation R in equation D.3 becomes approximated to

$$R \approx \begin{bmatrix} 1 & -\Theta_Z & \Theta_Y \\ \Theta_Z & 1 & -\Theta_X \\ -\Theta_Y & \Theta_X & 1 \end{bmatrix} \tag{D.7}$$

The 3D velocity at time t of the point $\mathbf{X}$ is calculated during small time interval Δt:

$$\dot{\mathbf{X}}(t) = \lim_{\Delta t \to 0} \frac{\mathbf{X}(t+\Delta t) - \mathbf{X}(t)}{\Delta t} \tag{D.8}$$

$$= \lim_{\Delta t \to 0} \frac{R\mathbf{X}(t) + T - \mathbf{X}(t)}{\Delta t} \tag{D.9}$$

$$= \lim_{\Delta t \to 0} \frac{(R\mathbf{X}(t) - I)\mathbf{X}(t) + T}{\Delta t} \tag{D.10}$$

$$= \lim_{\Delta t \to 0} \left(\frac{(R\mathbf{X}(t) - I)}{\Delta t} \mathbf{X}(t) + \frac{T}{\Delta t} \right) \tag{D.11}$$

$$= \Omega\mathbf{X}(t) + K \tag{D.12}$$

where $K = [K_X, K_Y, K_Z]^T$ is the translational velocity vector with $K_X = \dot{T_X}$, $K_Y = \dot{T_Y}$ and $K_Z = \dot{T_Z}$ the translational veloctity along the X, Y and Z axis respectively. Matrix Ω is the 3D angular velocity matrix defined by

$$\Omega = \lim_{\Delta t \to 0} \frac{(R\mathbf{X}(t) - I)}{\Delta t} \tag{D.13}$$

$$= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \begin{bmatrix} 0 & -\Theta_Z & \Theta_Y \\ \Theta_Z & 0 & -\Theta_X \\ -\Theta_Y & \Theta_X & 0 \end{bmatrix} \tag{D.14}$$

$$= \begin{bmatrix} 0 & -\Omega_Z & \Omega_Y \\ \Omega_Z & 0 & -\Omega_X \\ -\Omega_Y & \Omega_X & 0 \end{bmatrix} \tag{D.15}$$

where $\Omega_X = \dot{\Theta_X}$, $\Omega_Y = \dot{\Theta_Y}$ and $\Omega_Z = \dot{\Theta_Z}$ are the rotational velocities about the X, Y and Z axis respectively. To simplify the notation, the time dependency is omitted and the 3D velocity is formulated as

$$\dot{\mathbf{X}} = \Omega\mathbf{X} + K \tag{D.16}$$

Appendix E

Personal publications

Journal papers

- E. Corvee, S.A. Velastin and G.A. Jones. "Occlusion Tolerant Tracking using Hybrid Prediction Schemes". In *Acta Automatica Sinica, Special Issue on Visual Surveillance of Dynamic Sc*, 23(3), pages 356-369, 2003
- A. Toniappa, S. A. Barman, E. Corvee, M. J. Moseley, K. Cocker and A.R. Fielder. "Image quality assessment in retinal images of premature infants taken with the RetCam 120 digital fundus camera" in *The Imaging Science Journal*, 53(1), pages 51-59, March 2005

Conference paper

- S.A. Barman, A. Toniappa, E. Corvee and C. Sinthanayothin. "Classification of haemorrhage pathologies on digital fundus images using a combination of neural network and tracking algorithms" in *The 2nd ECTI Annual Conference (ECTI-CON 2005)*, Asia-Pattaya Beach Hotel, Thailand, May 12-13 2005

Bibliography

[1] G. Adiv. "Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] T. Altinalev, A. E. Cetin, and Y. Yardimci. Image registration techniques for multimodal sensors. *Proceedings of SPIE, SIgnal and Data Processing of Small Targets*, 4728, October 2002.

[3] P. Anandan. "Measuring Visual Motion from Image Sequences". *PhD thesis, COINS Dept., Amherst, Massachussets, USA, March*, 1987.

[4] P. Anandan, J. R. Bergen, K. J. Hanna, and R. Hingorani. "Hierarchical Model-Based Motion Estimation". *Proc. 2nd European Conference on Computer Vision (G. Sandini, ed.)*, 588:237–252, 1992.

[5] A. Anzalone and A. Machi. "Video-based Management of Traffic Light at Pedestrian Road Crossing". *Advanced Video-Based Surveillance Systems. C. S. Regazzoni, G. Fabri, G. Vernazza. Kluwer Academic Publishers*, pages 49–57, 1999.

[6] S. Araki, T. Matsuoka, N. Yokoyo, and H. Takemura. Real-time tracking of multiple moving object contours in a moving camera image sequence. *In IEICE Trans. on Information and Systems, Nagoya, Japan*, E83-D(7):1583–1591, July 2000.

[7] M. A. AUdette, F. P. Ferrie, and T. M. Peters. An algorithmic overview of surface registration techniques for medical imaging. *Medical Image Analysis*, pages 1–18, December 1999.

[8] A. Bab-Hadiashar and D. Suter. "Motion Segmentation: A Robust Approach". *IEEE International Workshop on the Interpretation of Visual Motion IVM'98, Santa Barbara, California*, pages 3–9.

[9] R. Venkatesh Babu and K. R. Ramakrishnan. Sprite generation from mpeg video using motion information. *International Journal on Image Graphics*, 4(2):263–280, 2004.

[10] A. Bainbridge-Smith and R.G. Lane. "Determining optical flow using a differential method". *Image and Vision Computing*, 15:11–22, 1997.

[11] A. Bakowski and G.A. Jones. "Video Surveillance Tracking using Colour Adjacency Graphs". In *IEE Conference on Image Processing and its Applications*, pages 794–798, Manchester, July 12-15 1999.

[12] Y. Bar-Shalom and T. Fortmann. *"Tracking and Data Association"*. Mathematics in Science and Engineering. Academic Press, 1988.

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

[14] A. Baumberg. "Reliable Feature Matching across Widely Separated Views". *IEEE Conference on Computer Vision and Pattern Recognition, Proceedings, Hilton Head Island, SC, USA, ISBN:0-7695-0662-3*, 1:774–781, 2000.

[15] S.S Beauchemin and J.L. Barron. "The Computation of Optical Flow". *ACM Computing Surveys*, 27(3):433–467, 1995.

[16] S.S Beauchemin and J.L. Barron. "On Discontinuous Optical Flow". *Computers And Artificial Intelligence*, 19(3):255–283, 2000.

[17] M. J. Black. "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields". *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[18] M. J. Black and A. Rangarajan. "On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision". *International Journal of Computer Vision*, 19(1):57–92, 1996.

[19] M.J. Black, Y. Yacoob, A.D. Jepson, and D.J. Fleet. "Learning Parameterized Models of Image Motion". *Computer Vision and Pattern Recognition (CVPR'97)*, O(3):561–567, 1997.

[20] P. Bouthemy. "A Maximum Likelihood Framework for Determining Moving Edges". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):499–511, 1989.

[21] P. Bouthemy and E. Francois. "Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequence". *International Journal of Computer Vision*, 10(2):157–182, 1993.

[22] S. A. Brock-Gunn, G. R. Dowling, and T. J. Ellis. "Tracking using Colour Information". *In Proc. ICARCV'94*, pages 686–690, 1994.

[23] M. J. Brooks, W. Chojnacki, and A. van de Hengel. "Fundamental matrix from optical flow: optimal computation and reliability evaluation". *Journal of Electronic Imaging*, 9(2):194–202, April 2002.

[24] Lisa G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4), December 1992.

[25] R.G. Brown and P.Y.C. Hwang. *"Introduction to Random Signals and Applied Kalman Filtering, 3d edition"*. John Wiley and Sons, 1985.

[26] E. Bruno and D. Pellerin. Video structuring, indexing and retrieval based on global motion wavelet coefficients. In *ICPR02*, pages III: 287–290, 2002.

[27] E. Bruno and D. Pellerin. "Global motion model based on bspline wavelets : application to motion estimation and video indexing". *In Proc. of the 2nd Int. Symposium. on Image and Signal Processing and Analysis, ISPA'01*, June 2001.

[28] H. Bunke and O. Bunke. *"Nonlinear Regression, Functional Relations and Robust Methods, Statistical Methods of Model Building, Volume 2"*. John Wiley and Sons, 1990.

[29] P.J. Burt and E.H. Adelson. "The Laplacian Pyramid As A Compact Image Code". *IEEE Transactions on Communication*, 31:532–540, 1983.

[30] W.E.L. Grimson C. Stauffer. "Adaptive background mixture models for real-time tracking". *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 246–252, 1999.

[31] Ting-Hsun Chang and Shaogang Gong. "Bayesian Modality Fusion for Tracking Multiple People with a Multi-Camera System". In *2nd European Workshop on Advanced Video-based Surveillance Systems*, pages 79–87, Kingston, UK, September 4 2001.

[32] L-F Chen, H-Y M Liao, and J-C Lin. "Wavelet-Based Optical Flow Estimation". *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):1–12, 2002.

[33] T. M. Chin, W. C. Karl, and A. S. Willsky. "A Maximum Likelihood Framework for Determining Moving Edges". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6):773–788, 1994.

[34] W.J. Christmas. "Spatial Filtering Requirements for Gradient-Based Optical Flow Measurement". *British Machine Vision Conference*, 1998.

[35] J. Clarke, S. Carlsson, and A. Zisserman. Detecting and tracking linear features efficiently. *Proc. 7th British Machine Vision Conf., Edinburgh, BMVA*, pages 415–424, 1996.

[36] C. Clifford, K. Langley, and D.J. Fleet. Centre-frequency adaptive iir temporal filters for phase-based image velocity estimation. In *IEE International Conference on Image Processing and Applications*, pages 173–178, 1995.

[37] I. Cohen and I. Herlin. "Non Uniform Multiresolution Method for Optical Flow and Phase Portrait Models: Environmental Applications". *International Journal of Computer Vision*, 33(1):29–49, 1999.

[38] J. Cooper and R. Hastings. "Uncertainty in the optical flow using the Kalman filter". *in Fourth International Conference on Control, Automation, Robotics and Vision*, 3:1681-1686, Dec. 1996.

[39] R. Cucchiara, C. C. Grana, M. Piccardi, and A. Prati. "Detecting moving objects, ghosts, and shadows in video streams". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, October 2003.

[40] T. Darrell. Integrated person tracking using stereo, color, and pattern detection. *ComputerVision and Pattern Recognition'98, Santa Barbara, CA*, pages 601–608, June 1998.

[41] D. DeCarlo and D. Metaxas. "Deformable Model-Based Shape and Motion Analysis from Images using Motion Residual Error". *Proceedings ICCV*, pages 113–119, 1998.

[42] S. L. Dockstader and T. Murat. Multiple camera tracking of interating and occluded human motion. *Proceedings of the IEEE*, 89(10):1441–1455, 2001.

[43] E. Dubois and J. Konrad. *"Estimation of 2-D Motion Fields from Image Sequences with Aplication to Motion-Compensated Processing In Motion Analysis and Image Sequence Processing chapter 3.8".* Kluwer Acad. Publication, 1993.

[44] F. Dufaux and F. Moscheni. "Background Mosaicking for Low Bit Rate Video Coding". *In Proc. ICIP, Lausanne, Switzerland*, 1996.

[45] Y. Dufournaud, C. Schmid, and R. Horaud. "Institut National de Recherche en Informatique et en Automatique". *CVIU*, 2003.

[46] T. Ellis and M. Xu. "Object Detection and Tracking in an Open and Dynamic World". In *Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Hawaii, December 2001.

[47] Mark Everingham and Barry Thomas. Supervised segmentation and tracking of non-rigid objects using a mixture of histograms model. *In 8th IEEE International Conference on Image Processing (ICIP2001)*, pages 62–65, 2001.

[48] de With P.H.N. Effelsberg W. Farin, D. Minimizing mpeg-4 sprite coding-cost using multi-sprites. *In: Proceedings Electronic Imaging 2004. Hrsg.: Panchanathan, S. and Vasudev, B./SPIE (The International Society for Optical Engineering). Bellingham, WA, USA: SPIE, 2004, S. , (Proceedings of SPIE)*, 5308, 2004.

[49] C. Fermuller, P. Baker, and Y. Aloimonos. "Visual Space-Time Geometry and Statistics". *British Machine Vision Conference 2002 (BMVA 2002)*, 1:1–42, 2002.

[50] Matthieu Ferrant, Simon K. Warfield, Charles R. G. Guttmann, Robert V. Mulkern, Ferenc A. Jolesz, and Ron Kikinis. 3d image matching using a finite element based elastic deformation model. In *MICCAI*, pages 202–209, 1999.

[51] Matthieu Ferrant, Simon K. Warfield, Arya Nabavi, Ferenc A. Jolesz, and Ron Kikinis. Registration of 3d intraoperative MR images of the brain using a finite element biomechanical model. In *MICCAI*, pages 19–28, 2000.

[52] G. L. Foresti and C. Micheloni. a robust feature tracker for active surveillance of outdoor scenes. *Electronic Letters on Computer Vision and Image Analysis*, 1(1):21–34, 2003.

[53] H. Foroosh, J. B. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Trans. Image Processing (issue 3)*, 11(3):188–200, 2002.

[54] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Proceedings 13. Conf. on Uncertainty in Artificial Intelligence*, pages 175–181, 1997.

[55] L. M. Fuentes and S. A. Velastin. "People Tracking in Surveillance Applications". *Proc. 2nd IEEE International Workshop on PETS,Hauai, Hawai, USA, December 9 2001*, 2001.

[56] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving feature tracking with robust statistics. *Pattern Analysis and Applications*, 2:312–320, 1999.

[57] T. Gautama and M.M Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Trans. Neural Networks*, 13(5):1127–1136, 2002.

[58] P.R. Giaccone. Motion analysis of cinematographic image sequences, 2000.

[59] A. Giachetti. "Matching techniques to compute image motion". *Image and Vision Computing*, 18:247–260, 2000.

[60] R. Hanson, J. Stutz, and P. Cheeseman. Baysian classification theory. *Technical Report FIA-90-12-7-01, NASA Ames Research Center, Artificial Intelligence Brand*, May 1991.

[61] R. M. Haralick and J. S. Lee. "The facet approach tooptical flow". *Proc. of Image Understanding Workshop (science Applications) Arlington, VA, December*, 1983.

[62] C. Harris and M. Stephens. "The facet approach tooptical flow". *Proceedings of The Fourth Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[63] H. Haussecker, C. Garbe, H. Spies, and B. Jahne. "A total least Squares Framework for Low-Level Analysis of Dynamic Scenes and Processes". *DAGM,Bonn, Germany*, pages 240–249, 1999.

[64] H. W. Hausseker and D. J. Fleet. "Computing Optical Flow with Physical Models of Brightness Variation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), 2001.

[65] J. Heuer and A. Kaup. "Global motion estimation in image sequences using robust motion vector field segmentation". *Proceedings of the seventh ACM international conference on Multimedia (Part 1), Orlando, Florida, United States*, 1(ISBN:1-58113-151-8):261–264, 1999.

[66] L. Hill and T. Vlachos. "Motion measurement using adaptive phase correlation". In *Electronics Letters*, 6th December 2001.

[67] B.K.P. Horn. *"Robot Vision"*. MIT Press, 1986.

[68] B.K.P. Horn and B.G. Schunck. "Determining Optical Flow". *Artificial Intelligence*, 17:185–203, 1981.

[69] M. Hotter and R. Thoma. "Image Segmentation based on Object Oriented Mapping Parameter Estimation". *Signal Processing*, 15:315–334, 1988.

[70] Y. Huang, X. Zhuang, and J.E. Cavanaugh. "Optic Flow Field Segmentation and Motion Estimation Using a Robust Genetic Partitioning Algorithm". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1177–1190, 1995.

[71] D. Harwood I. Haritoaglu and L. Davis. W4: Who? when? where? what? a real time system for detecting and tracking people. *IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.

[72] I.Haritaoglu, D.Harwood, and L.S.Davis. "A Real Time System for Detecting and Tracking People". *Image and Vision Computing Journal*, January 1999.

[73] I.Haritaoglu, D.Harwood, and L.S.Davis. "W4: Real-time Surveillance of people and their Activities". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

[74] G.A. Jones. "Matching Corner Features using Edge Contour Data". *Technical Report KUCSES-97-02*, 1997.

[75] D.J. Thirde G.A. Jones J.R. Renno, J. Orwell. "Shadow Classification and Evaluation for Soccer Player Detection". In *British Machine Vision Conference, BMVA - ISBN/ISSN 1-901725-25-1*, pages 839–848, Kingston upon Thames, September 7-9 2004.

[76] Frederic Jurie and Michel Dhome. A simple and efficient template matching algorithm. *Eigth International Conference on Computer Vision (ICCV'01)*, 2:544–549, 2001.

[77] Frederic Jurie and Michel Dhome. Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–1000, 2002.

[78] P. KaewTraKulPong and R. Bowden. "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection". *In Proc. 2nd European Workshop*

on Advanced Video Based Surveillance Systems, AVBS01, Sept 2001, VIDEO BASED SURVEILLANCE SYSTEMS: Computer Vision and Distributed Processing, 2001.

[79] A. Katsaggelos. A multiple input image restoration approach. *J. Visual Commun. Image Represent.*, pages 93–103, Sept. 1990.

[80] J. K. Kearney, W. B. Thompson, and D. L. Boley. "Optical Flow Estimation: An Error Analysis of Gradient-Based Methods with Local Optimization". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2), 1987.

[81] S. Khan and M. Shah. "Tracking People in Presence of Occlusion". *Asian Conference on Computer Vision. Taipei, Taiwan*, 2000.

[82] A. Kokaram and P. Delacourt. "A new global motion estimation algorithm and its application to retrieval in sports events". *IEEE workshop on Multimedia Signal Processing. MMSP01, France*, 2001.

[83] D. Koller, K. Daniilidis, and H. H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[84] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Proceedings of the International Conference on Pattern Recognition*, pages 126–131, Israel, 1994.

[85] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *ECCV (1)*, pages 189–196, 1994.

[86] Stefan Kruger and Andrew Calway. Image registration using multiresolution frequency domain correlation. In *British Machine Vision Conference*, pages 316–325. British Machine Vision Association, 1998.

[87] H. Liu, T-H. Hong, M. Herman, and R. Chellapa. "Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms". *Proc. Fourth European Conference on Computer Vision*, 2:174–183, 1996.

[88] L. Marcenaro, L. Marchesotti, and C.S. Regazzoni. "Tracking and Counting Multiple Interacting People in Indoor Scenes". In *Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 56-61, Copenhagen, June 1 2002.

[89] V. Markandey and B. E. Flichbaugh. "Multispectral Constraints for Optical Flow Computation". *Proc. of Tird IEEE International Conference on Computer Vision ICCV, Osaka, Japan*, pages 38–41, 1990.

[90] J. Matas and O. Chum. Randomized ransac. *J. Visual Commun. Image Represent.*, Proceedings of the CVWW'02, Wien, Austria:49–58, February 2002.

[91] Stephen J. McKenna, Yogesh Raja, and Shaogang Gong. Object tracking using adaptive color mixture models. In *ACCV (1)*, pages 615–622, 1998.

[92] A. Mitichie, Y.F. Wang, and J.K. Aggarwal. "Experiments in computing optical flow with the gradient-based, Multiconstraint method". *Pattern Recognition*, 20(2):173–179, 1987.

[93] A. Mittal and D. Huttenlocher. "Scene Modeling for Wide Area Surveillance and Image Synthesis". *Computer Vision and Pattern Recognition (CVPR'00), June 13-15, 2000*, 2:2160, 2000.

[94] D. Murray and A. Basu. "Motion Tracking with an Active Camera". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):449–459, 1994.

[95] A. Lippman N. Vasconcelos. "Empirical Bayesian EM-based Motion Segmentation". *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 527–532, 1997.

[96] H.H. Nagel. "On the Estimation of Optical Flow: Relations Between Different Approaches and Some New Results.". *Artificial Intelligence*, 33(3):299–324, 1987.

[97] H.H. Nagel. "On a Constraint Equation for the Estimation of Displacement Rates in Image Sequences". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1), 1989.

[98] S. Negahdaripour. "Revised Definition of optical Flow: Integration of Radiometric and Geometric Cues for Dynamic Scene Analysis". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9), 1998.

[99] S. Negahdaripour and K.P. Horn. "Direct Passive Navigation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):168–176, 1987.

[100] F. Odone, A. Fusiello, and E. Trucco. Robust motion segmentation for content-based video coding, 2000.

[101] J. Orwell, P. Remagnino, and G.A. Jones. "From Connected Components to Object Sequences". In *First IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 72–79, 2000.

[102] Justus H. Piater, Stephane Richetto, and James L. Crowley. "Event-based Activity Analysis in Live Video using a Generic Object Tracker". In *Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–8, Copenhagen, June 1 2002.

[103] W. H. Press, B. P. Flannery, W. T. Vetterling, and S. A. Teukolsky. *"Numerical Recipes in C: The Art of Scientific Computing"*. Cambridge University Press, Cambridge (UK) and New York.

[104] T. Kanadeand H. Fujiyoshi D. Duggins R. Collins, A. Lipton and Y. Tsin. "A System for Video Surveillance and Monitoring: VSAM Final Report". *Carnegie Mellon University. CMU-RI-TR-00-12*, 2000.

[105] J. Renno, J. Orwell, and G.A. Jones. "Towards Plug-and-Play Visual Surveillance: Learning Tracking Models". In *IEEE International Conference on Image Processing*, page Accepted for Publication, Rochester, New York, September 22-25 2002.

[106] J.R. Renno, J. Orwell, and G.A. Jones. "Learning Surveillance Tracking Models for the Self-Calibrated Ground Plane". In *British Machine Vision Conference*, Cardiff, September 2002.

[107] C. Ridder, O. Munkelt, and H. Kirchner. "Adaptive background estimation and foreground detection using Kalman filtering". *In Proc. Int. Conf. on recent Advances in Mechatronics, ICRAM'95, UNESCO Chair of Mechatronic*, pages 193–199, 1995.

[108] Romer Rosales and Stan Sclaroff. "Improved Tracking of Multiple Humans with Trajectory Predcition and Occlusion Modeling". *Proceedings IEEE Conf. on CVPR, Workshop on the Interpretation of the Visual World, CA, Santa Barbara*, 2, 1998.

[109] P. L. Rosin and T. Ellis. "Image difference threshold strategies and shadow detection". *Proc. British Machine Vision Conference BMVC 1995, Birmingham, UK, September 95*, pages 347–356.43, 1995.

[110] P.J. Rousseeuw and A.M. Leroy. *"Robust Regression and Outlier Detection"*. John Wiley and Sons, 1987.

[111] E. Saez, J. M. Palomares, J. I. Benavides, and N. Guil. "global motion estimation algorithm for video segmentation". *Proceedings of Visual Communications and Image Processing 2003*, 2003.

[112] R.J. Schalkoff. *"Digital Image Processing And Compuetr Vision"*. John Wiley and sons, inc., 1989.

[113] Andrew Senior. "Tracking People with Probabilistic Appearance Models". In *Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 48–55, Copenhagen, June 1 2002.

[114] L.S. Shapiro. *"Affine Analysis of Image Sequences, chapter 6"*. Cambridge University Press, NY, USA, 1995.

[115] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[116] N.T. Siebel and S.J. Maybank. "Real-time Tracking of Pedestrians and Vehicles". In *Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Hawaii, December 2001.

[117] S.M. Smith and J.M. Brady. "ASSET-2: Real-time motion segmentation and shape tracking". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):814–820, 1995.

[118] A. Smolic, J.-R. Ohm, and T. Sikora. "Direct Estimation of Long-term Global Motion Parameters using Affine and Higher Order Polynomial Models for Dynamic Sprite Coding". *Picture Coding Symposium '99*, April 1999.

[119] A. Smolic, J.-R. Ohm, and T. Sikora. "Object-Based Global Motion Estimation Using a Combined Feature Matching and Optical Flow Approach". *Proc. VLBV'98 Workshop, Chicago*, October 1998.

[120] Martin Spengler and Bernt Schiele. Towards robust multi-cue integration for visual tracking. *Lecture Notes in Computer Science*, 2095:93, 2001.

[121] E. V. Stansfield. "Introduction to Kalman filters". In *UK Adaptive Signal Processing Club Discussion Meeting*, The Racal Training Services Conference Centre, Heckfield Place near Reading, Berkshire, UK, 7th March 2001.

[122] C. Stauffer and W.E.L. Grimson. "Learning Patterns of Activity using Real-Time Tracking". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[123] C. Stiller, J. Konrad, and R. Bosch. "Estimating Motion in Image Sequences, A tutorial on modelling and computation of 2D motion". *IEEE Signal Processing Magazine*, pages 70–91, 1999.

[124] C. Sun. "Fast Optical Flow Using Cross Correlation and Shortest-Path Techniques". In *DICTA99*, pages 143–148, 1999.

[125] C. Sun and S. Pallotino. "Circular Shortest Path on Regular Grids". *In ACCV2002: the 5th Asian Conference on Computer Vision, Melbourne, Australia*, pages 852–857, 23-25 Janvier 2002.

[126] R. Szeliski. Video mosaics for virtual environment. *IEEE Computer Graphics and Applications (c) 1996 IEEE*, 16(2), 1996.

[127] Geoffrey Taylor and Lindsay Kleeman. Fusion of multimodal visual cues for model-based object tracking. *Proceedings of the 2003 Australasian Conference on Robotics and Automation, Brisbane, Australia*, December 1-3 2003.

[128] D. Thirde. Efficient supervised depth content generation in the context of video frames and video sequences. Technical Report Transfer Report, Kingston University, School of Computing and Information System, 2002.

[129] M. Tistarelli. "Multiple Constraints to Compute Optical Flow". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1243–1250, 1996.

[130] B. Tom and A. Katsaggelos. Resolution enhancement of video sequences using motion compensation. *in Proceedings of the IEEE International Conference on Image Processing, Lausanne, Switzerland*, I:713–716, Sept 1996.

[131] P. H. S. Torr, P. A. Beardsley, and D. W. Murray. Robust vision. In J. Illingworth, editor, *bmvc94*, pages 145–155. BMVA Press, 1994.

[132] O. Tretiak and L. Pastor. "Velocity estimation from image sequences with second order differential operators". *In Proc. Seventh International Conference on Patte Recognition, Montreal, Canada, July*, pages 16–19, 1984.

[133] D. Tweed and A. Calway. "Motion Segmentation Based on Integrated Region Layering and Motion Assignment". *In Proceedings of Fourth Asian Conference on Computer Vision*, pages 1002–1007, January 2000.

[134] D. Tzovaras, N. Grammalidis, and M. G. Strintzis. "Three-dimensional camera motion estimation and foreground/background separation for stereoscopic image sequences". *Society of Photo-Optical Instrumentation Engineers*, 36(2):574–579, 1997.

[135] J. Weber and J. Malik. "Rigid Body Segmentation and Shape Description from Dense optical Flow under Weak Perspective". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):139–143, 1997.

[136] Joseph Weber and Jitendra Malik. Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14(1), 1995.

[137] Q. Wei, Y. Zhong, and C-J. Zhang. "A New Robust Global Motion Estimation Approach Used in MPEG-4". *Journal of Tsinghua University Science and Technology*, pages 1–13, January 2000.

[138] Y. Weiss. "Smoothness in Layers: Motion Segmentation Using Nonparametric Mixture Estimation". *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 520–526, 1997.

[139] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[140] R. J. Woodham. "Multiple light source optical flow". *Proc. of third IEEE International Conference on Computer Vision (ICCV), Osaka, Japan*, pages 42–46, 1990.

[141] L. Zhang, T. Sakurai, and H. Miike. "Detection of motion fields under spatio-temporal non-uniform illumination". *Image and Vision Computing*, 17:309–320, 1999.

[142] L. Zhi, W. Chao, and Z. Hong. "A new registration of interferometric SAR: Least-squares registration". In *22nd Asian Conference on Remote Sensing*, Singapore, 5-9 November 2001.

[143] A. Zomet and S. Peleg. Efficient Super-Resolution and applications to mosaics. *Proceedings 15th Int. Conf. on Pattern Recognition, Barcelona, Spain*, 1(ISBN: 0-7695-0750-6):579–583, 2000.