

Rigid Structure from Video

Pedro Manuel Quintas Aguiar
(Mestre)

Dissertação para a obtenção do Grau de Doutor em
Engenharia Electrotécnica e de Computadores

Orientador Científico: Doutor João José dos Santos Sentieiro
Co-Orientador Científico: Doutor José Manuel Fonseca de Moura

Júri: Reitor da Universidade Técnica de Lisboa
Doutor A. Murat Tekalp, University of Rochester
Doutor José Manuel Fonseca de Moura, Carnegie Mellon University
Doutor Hélder de Jesus Araújo, FCT, Universidade de Coimbra
Doutor Fernando Eduardo Rebelo Simões, IST, UTL
Doutor João José dos Santos Sentieiro, IST, UTL
Doutor Fernando Manuel Bernardo Pereira, IST, UTL
Doutor João Paulo Salgado Arriscado Costeira, IST, UTL

Janeiro 2000

Abstract

This thesis addresses two problems in content-based video analysis: the segmentation of a 2D rigid moving object; and the inference of 3D rigid structure.

In Part I, we consider motion segmentation. Existing methods often fail to detect the motions of low textured regions and regions moving against low contrast background. We describe an algorithm to segment low textured moving objects that move against a low contrast background. Our approach has two distinguishing features. Our algorithm processes all frames available, as needed, while the majority of current motion segmentation methods use only two or three consecutive images. Second, we model explicitly the occlusion of the background by the moving object and recover the shape of the moving object directly from the image intensity values. This contrasts with other approaches that deal with low textured scenes by attempting to smooth out a sparse set of motion measurements.

In Part II, we develop a factorization method that recovers the 3D shape and 3D motion of rigid moving objects whose surface shape is parameterized by a finite set of parameters. Our approach induces a parametric model for the 2D motion of the brightness pattern in the image plane. To estimate the 3D shape and 3D motion parameters from the 2D motion parameters, we introduce the *surface-based rank 1 factorization* algorithm. Our method uses an appropriate linear subspace projection that leads to the factorization of a matrix that is rank 1 in a noiseless situation. This allows the use of fast iterative algorithms to compute the 3D structure that best fits the data. We track regions where the 2D motion in the image plane is described by a single set of parameters. Our method contrasts with the original *factorization method* that required tracking a large number of pointwise features, in general a difficult task, and the factorization of a rank 3 matrix.

We apply to both problems *Maximum Likelihood* estimation techniques. Illustrative examples show the good quality of our algorithms.

Keywords: Rigid Structure from Motion, Surface-based Rank 1 Factorization, Motion Segmentation, Motion Analysis, Video Sequence Processing, Maximum Likelihood.

Resumo

Esta tese trata dois problemas de análise de sequências de vídeo: a segmentação de objectos rígidos 2D em movimento; e a inferência de estrutura rígida 3D.

A parte I é dedicada à segmentação baseada em movimento. Os métodos existentes falham frequentemente na detecção do movimento de regiões de textura uniforme e de regiões que se movem sobre um fundo de baixo contraste. Na parte I, desenvolve-se um algoritmo que segmenta objectos de textura uniforme que se movem sobre fundo de baixo contraste. A técnica proposta tem duas características próprias que a distinguem das existentes. Primeiro, o algoritmo desenvolvido processa todas as imagens disponíveis, como necessário, enquanto a maioria dos métodos de segmentação baseada em movimento usam apenas duas ou três imagens consecutivas. Segundo, modeliza-se explicitamente a oclusão do fundo pelo objecto em movimento e estima-se a forma deste directamente a partir dos valores de intensidade das imagens. Esta particularidade contrasta com outros métodos que, para lidar com cenas de textura uniforme, tentam suavizar um conjunto esparsa de medidas de movimento.

Na parte II, desenvolve-se um método de factorização para inferir forma 3D e movimento 3D de objectos rígidos cuja superfície é parameterizada por um conjunto finito de parâmetros. A abordagem proposta induz um modelo paramétrico para o movimento 2D do padrão de intensidade luminosa no plano da imagem. Para estimar os parâmetros de forma 3D e movimento 3D a partir dos parâmetros de movimento 2D, desenvolve-se o algoritmo de *factorização de característica 1 baseada em superfícies*. Este método usa uma projecção num subespaço linear que leva à factorização de uma matriz de característica 1 na ausência de ruído. Deste modo, podem-se usar algoritmos iterativos rápidos para calcular a estrutura 3D que melhor se ajusta às observações. A técnica proposta faz o seguimento de regiões onde o movimento 2D no plano da imagem é descrito por um único conjunto de parâmetros. O método desenvolvido contrasta com o *método da factorização* original que requer o seguimento de um elevado número de regiões elementares (pontuais), em geral uma tarefa difícil, e a factorização de uma matriz de característica 3.

Aplicam-se técnicas de estimação de *Máxima Verosimilhança* a ambos os problemas. A qualidade dos algoritmos propostos é demonstrada através de exemplos ilustrativos.

Palavras-chave: Estrutura Rígida a partir do Movimento, Factorização baseada em Superfícies, Segmentação baseada em Movimento, Análise de Movimento, Processamento de Sequências de Vídeo, Máxima Verosimilhança.

e-acknowledgements

To moura@ece.cmu.edu, jjss@isr.ist.utl.pt.

To {rsj,asif,martins,cjc}@ece.cmu.edu.

To {fmg,simoes}@isr.ist.utl.pt, {bioucas,mtf,jleitao}@lx.it.pt.

To {jpc,mmv}@cs.cmu.edu, {cb8t,ls49,aa3f}@andrew.cmu.edu, teix@pitt.edu.

To ISR, IST, CMU, INVOTAN, FCG. Also to



aguiar@{isr.ist.utl.pt,ece.cmu.edu}

Contents

1	Introduction	13
1.1	2D and 3D Content-Based Video Representations	14
1.2	Maximum Likelihood Inference of Rigid Structure from Video	18
1.3	Thesis Overview	24
I	Segmentation of 2D Moving Object	31
2	Statement of the Segmentation Problem	33
2.1	Introduction	33
2.2	Related Work	34
2.3	Notation	36
2.4	Problem Formulation	39
2.5	Maximum Likelihood Estimation	42
2.6	Summary	44
3	Image Motion Estimation	45
3.1	Introduction	45
3.2	Motion Estimation	48
3.3	Estimation Error	52
3.4	Translational Motion	55
3.5	Affine Motion	63
3.6	Summary	68
4	Direct Inference of 2D Rigid Shape	69
4.1	Introduction	69
4.2	Initialization: Object Mask	71
4.3	Minimization Procedure: Two-Step Iterative Algorithm	73
4.4	Experiment	80
4.5	Summary	83

5	Analysis of the Segmentation Algorithm	85
5.1	Introduction	85
5.2	Statistics of the Segmentation Matrix	86
5.3	Bounds on the Probability of Error	91
5.4	Experiment	95
5.5	Conclusions	97
6	Real Video Experiments	101
6.1	Robot Soccer	101
6.2	Road Traffic	104
II	Inference of 3D Rigid Structure	107
7	3D Structure from 2D Video	109
7.1	Introduction	109
7.2	Related Work	110
7.3	Problem Formulation	113
7.4	Maximum Likelihood Estimation	115
7.5	Structure from Motion: Approximate Maximum Likelihood Estimate . . .	118
7.6	Summary	122
8	Rank 1 Factorization	123
8.1	Introduction	123
8.2	Factorization Approach	125
8.3	Rank 1 Factorization	130
8.4	Normalization Failure	135
8.5	Rank 1 Approximation: Iterative Algorithm	139
8.6	Experiments	141
8.7	Summary	147
9	Surface-Based Factorization	149
9.1	Introduction	149
9.2	Example	150
9.3	Surface-Based Factorization	155
9.4	Weighted Factorization	162
9.5	Experiments	165
9.6	Summary	172

10 Direct Inference of 3D Rigid Shape	175
10.1 Introduction	175
10.2 Image Sequence for Known Motion	177
10.3 Minimization Procedure: Continuation Method	180
10.4 Experiment	182
10.5 Summary	186
11 Real Video Experiments	189
11.1 Box	189
11.2 Pedestal	196
11.3 Clown	197
12 Conclusion	199
12.1 Thesis Summary	199
12.2 Major Contributions	205
12.3 Future Directions	206
A Moments of Functions of Random Variables	213
A.1 First-order approximation of the mean	213
A.2 First-order approximation of the covariance	214
Bibliography	217

Chapter 1

Introduction

The human capability of understanding visual data is impressive. We easily infer environmental characteristics from a single photographic image. Since the early days of image analysis and computer vision, scientists attempt to reproduce this capability on the computer. Appropriate image formation models and computer algorithms have been developed that analyze an image from inferences with respect to those models. These algorithms are useful when analyzing single images in terms of primitives like edges and textures. When the goal is the inference of three-dimensional (3D) structure, methods that use a single image are based on cues such as shading and defocus. These methods fail to give reliable 3D shape estimates for unconstrained real-world scenes. If no prior knowledge about the scene is available, very little can be inferred from a single image with respect to the 3D structure.

With the ever increase of computational power, attention has turned to the analysis of sequences of images. *Motion* is a powerful cue in an image sequence that is absent when single images are considered. The motion of the image brightness pattern contains significant information about the 3D structure of the scene. This thesis is about *motion analysis*, the task of analyzing an image sequence according to the motions present. In section 1.1, we start by motivating the specific problems addressed in the thesis, pointing out bibliographic references that inspired our work. Then, in section 1.2, we describe the overall approach to the recovery of rigid structure from video. Finally, in section 1.3, we

detail the organization of the thesis.

1.1 2D and 3D Content-Based Video Representations

This thesis addresses two problems within motion analysis. The first problem concerns the accurate segmentation of a two-dimensional (2D) rigid moving object. The second is the inference of three-dimensional (3D) rigid structure. We address these problems from the fundamental point of view of digital video representation [57].

The segmentation of 2D moving objects is motivated in the context of *Generative Video* (GV). GV is a framework for content-based video sequence representation, introduced by Jasinschi and Moura in references [35, 36]. In GV the operational units are not the individual images in the original sequence, as in standard methods, but rather the world images and the ancillary data. The world images encode the non-redundant information about the video sequence. They are augmented views of the world – background world image – and complete views of moving objects – figure world images. The ancillary data registers the world images, stratifies them at each time instant, and positions the camera with respect to the layering of world images. The world images and the ancillary data are the GV representation, the information that is needed to regenerate the original video sequence. We discuss the segmentation of 2D moving objects in the context of generating the world images and ancillary data for the GV representation of a video clip.

Just like an image is worth ten thousand words and video enhances tremendously our visual perception of the environment, 3D represents the next higher level in recreating a natural immersive multimedia environment for participants to interact collaboratively, and/or viewers to enjoy. The volume experience enables users to perceive differently the same scene from their own vantage point of view. In the original formulation of GV, world images are modeled as simple planar scenarios. This representation fails when the relative depth of the scene structure is not negligible. We generalize to 3D shaped scenarios the GV representation. This framework motivates the second problem addressed in the thesis

- the recovery of the 3D structure (3D shape and 3D motion) from a 2D video sequence.

2D motion segmentation in low texture and low contrast

An important task in building a GV representation of a video sequence is the automatic segmentation of the moving objects. Motion segmentation methods often fail to detect *low textured* regions and regions moving against a *low contrast* background. To appreciate this difficulty, we apply the algorithm of reference [23] to segmenting a low textured moving object. The two left images in Figure 1.1 show two consecutive frames of a video sequence. The right image of Figure 1.1 displays the template of the moving car reconstructed by excluding from the regions that changed between the two co-registered frames the ones that correspond to uncovered background areas, see reference [23]. The small regions, for example in the top left corner, that due to the noise are misclassified as belonging to the car template can be discarded by an adequate morphological post-processing. However, the car template is highly incomplete, with the regions in the interior of the car being misclassified as belonging to the background, due to the low texture of the car.



Figure 1.1: Motion segmentation in low texture.

In the thesis we focus on the segmentation of low textured objects moving against a low contrast background. We develop an algorithm that integrates across the frames in the sequence the intensity differences between the moving object and the background. Through this temporal integration, the algorithm recovers accurate templates of the moving objects, even when they move against a low contrast background.

Our approach is characterized by two distinguishing features. First, while the majority of the current motion segmentation methods use only two or three consecutive images,

our algorithm processes all frames available, as needed. Second, while other approaches deal with the incompleteness of the templates by attempting to smooth out a sparse set of motion measurements, our method recovers the shape of the moving object directly from the image intensity values.

Some authors cope with low textured scenes by attempting to smooth out a sparse set of motion measurements. Their approaches work by coupling motion-based segmentation with prior knowledge about the scene as in statistical regularization or by combining motion with other attributes. In general, these methods lead to complex and time consuming algorithms. Irani, Rousso, and Peleg [31, 32] proposed one of the few approaches using temporal integration to segment a moving object. Their method works by averaging the images registered according to the motion of the different objects in the scene. After processing a number of frames, each of these integrated images is expected to show only one sharp region corresponding to the tracked object. This region is found by detecting the stationary regions between the corresponding integrated image and the current frame. With low textured objects in low contrast background, this technique does not perform well. Our approach is related to this one. However, we use all the frames available rather than just a single frame to estimate the templates of the moving objects. We will see that, by integrating the small differences across several images, our technique resolves the difficulties that arise in low texture and low contrast video sequences.

Fast 3D rigid structure from video

The key step in generating automatically 3D model-based representations for video is the recovery of the 3D shape of the environment and the 3D motion of the camera from the video sequence. If no prior knowledge about the scene is available, the cue to estimating the 3D structure (3D shape and 3D motion) from a video sequence is the 2D motion of the brightness pattern in the image plane. For this reason, the problem is generally referred to as *structure from motion* (SFM). Early approaches to SFM processed a single pair of consecutive frames. Two-frame based algorithms are highly sensitive to image noise,

and, when the object is far from the camera, i.e., at a large distance when compared to the object depth, they fail even at low level image noise. More recent research has been oriented towards the use of longer image sequences. The problem of estimating 3D structure from multiple frames has a larger number of unknowns (the 3D shape and the set of 3D positions) but it is more constrained than the two-frame SFM problem because of the rigidity of the scene. Among the existing approaches to the multiframe SFM problem, the *factorization method* introduced by Tomasi and Kanade in references [59, 60, 61] is an elegant method to recover rigid structure from an image sequence. In references [59, 61], the 2D projection of a set of feature points is tracked along the image sequence. The 3D shape and motion are then estimated by factorizing a measurement matrix whose entries are the set of trajectories of the feature point projections. Tomasi and Kanade pioneered the use of linear subspace constraints in motion analysis. In fact, the key idea underlying the factorization method is the fact that the rigidity of the scene imposes that the measurement matrix lives in a low dimensional subspace of the universe of matrices. Tomasi and Kanade have shown that the measurement matrix is a rank 3 matrix in a noiseless situation. References [59, 61] use the orthographic projection model. The factorization method was later extended to the scaled-orthography and para-perspective models, see references [47, 48, 49], and to the multibody scenario, see references [19, 20, 21].

In this thesis, we use linear subspace constraints to solve SFM in the more general scenario of recovering 3D motion and a parameteric description of the 3D shape from a sequence of 2D motion parameters. Exploiting further the existing subspace constraints, we solve the SFM problem by factorizing a matrix that is rank 1 in a noiseless situation, rather than a rank 3 matrix as in the original factorization method.

To recover in an expedite way the 3D motion and the 3D shape, we introduce the *surface-based factorization*. Under our general scenario, we describe the shape of the object by a parameterization of its surface. We show that this parametric description of the 3D shape induces a parameteric model for the 2D motion of the brightness pattern in the image plane. The surface-based factorization approach overcomes a limitation

of the original factorization method of Tomasi and Kanade. Their approach relies on the matching of a set of features along the image sequence. To provide dense depth estimates, their method usually needs hundreds of features that are difficult to track and that lead to a complex correspondence problem. Instead of tracking pointwise features, our method tracks regions where the optical flow is described by a single set of parameters. Our approach avoids the correspondence problem and is particularly suited when there is good prior knowledge about the shape of the 3D objects. A good practical scenario is when constructing 3D models for buildings that are well described by piecewise flat surfaces.

The algorithm that we develop has a second major feature – its computational simplicity. By making an appropriate linear subspace projection, we show that the unknown 3D structure can be found by factorizing a matrix that is rank 1 in a noiseless situation. This contrasts with the factorization of a rank 3 matrix as in the original method of Tomasi and Kanade. This allows the use of faster iterative algorithms to compute the matrix that best approximates the data.

1.2 Maximum Likelihood Inference of Rigid Structure from Video

We formulate the problem of inferring rigid structure from video by using the analogy between the visual perception mechanism and a classical communication system, see Figure 1.2. This analogy has been used to deal with perception tasks involving a single image, see references [38] and [45].

In a communication system, see top of Figure 1.2, the transmitter receives the message S from a source and sends it to the receiver. The transmitter codes the message and sends the resulting signal I^* through the channel. The receiver gets the signal I at the output of the channel, a noisy version of the signal I^* . The receiver decodes I obtaining the estimate \hat{S} of the message S . In statistical communications theory, we describe statistically the channel distortion and design the receiver according to a statistically optimal

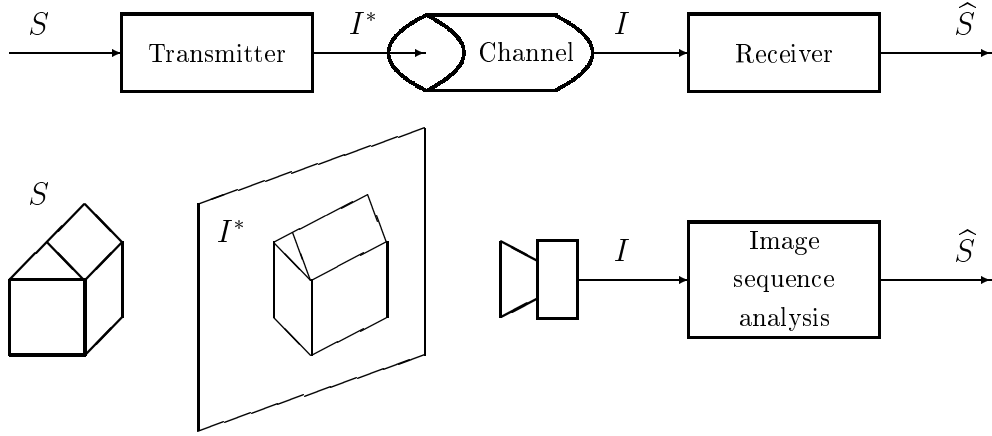


Figure 1.2: Communications system and image sequence analysis: analogy.

criteria. For example, we can estimate \hat{S} as the message S that maximizes the probability of receiving the signal I , conditioned on the message S sent. This is the *Maximum Likelihood* (ML) estimate. If a statistical description of the message source is available, this can be incorporated into the estimation by using a *Bayesian* framework.

The communication system is a good metaphor for the problem of recovering real world structure from video, represented at the bottom of Figure 1.2. The message source is the environment in the real world. The transmitter is the geometric projection mechanism that transforms the real world S into an ideal image I^* . The channel is the video camera that captures the image I , a noisy version of I^* . The receiver is the image sequence analysis system. The task of this system is to recover the reality that originated the image sequence that was captured.

The ultimate goal of the research work described in this thesis is to design an image sequence analysis system that computes in an expedite way the ML estimate of the real world environment. The joint ML estimation of the complete set of unknowns – *motions*, *shapes*, and *textures* (these terms will be defined in a precise way later on) – leads to the minimization of a complex cost function. The minimization of the ML cost function with respect to the complete set of unknowns is a highly complex task – in general its analytic solution can not be found. The large number of unknowns usually excludes standard

numerical iterative optimization techniques due to their high computational cost, and to their serious common problems with local minima. A major problem in this type of complex optimization is the unsupervised search for good initial guesses. Rather than blindly applying standard optimization methods, our approach is guided by the specific characteristics of each of the problems we address.

The remaining of this section outlines the methods we propose to accomplish the expedite minimization of the ML cost functions for the problems of two-dimensional (2D) motion segmentation and inference of three-dimensional (3D) structure. To derive computationally simple algorithms, we make approximations well supported by practical experience. These sub-optimal solutions to the ML estimate can be further improved if used as the initial guess for a standard iterative optimization method. We want to emphasize that our algorithms solve a key task in this type of complex optimization problems – the unsupervised search for good initial guesses.

Multiframe segmentation of a 2D moving object

To segment accurately a 2D moving object, we process a set of multiple frames. By modeling the rigidity of the moving object across time, we succeed where other methods fail – in segmenting low textured moving objects and objects moving against low contrast background.

According to the analogy illustrated in Figure 1.2, we accomplish the segmentation of a 2D moving object by computing the ML estimate of all the unknown parameters in the 2D image sequence model: the template of the moving object, the intensity level of the object pixels (object texture), the intensity level of the background pixels (background texture), the 2D motions of the object, and the 2D motions of the background (camera motion). We consider that the moving object is in the foreground of the scene in front of the background. The moving object is opaque and it is visible through the entire sequence.

The joint ML estimation of the complete set of unknown parameters is a complex

task. Motivated by our experience with real video sequences, we decouple the estimation of the motions (moving object and camera) from that of the remaining parameters. The motions are estimated on a frame by frame basis by using known methods. We will see that this procedure minimizes a simplified version of the overall ML cost function. Then, we introduce the motion estimates into the ML cost function and minimize this function with respect to the remaining parameters. The estimate of the object texture is obtained in closed form. To estimate the texture of the background and the moving object template, we develop a fast two-step iterative algorithm. The first step estimates the background for a fixed template – the solution is obtained in closed form. The second step estimates the template for a fixed background – the solution is given by a simple binary test evaluated at each pixel. The algorithm converges in a few iterations, typically three to five iterations.

Our algorithm recovers low textured moving objects taking into account the rigidity of the object over the set of multiple frames. Furthermore, our approach is well suited to the even more challenging situation where the object moves against a low contrast background because the ML cost function integrates across time the small intensity differences between the object and the background. The template estimated by minimizing the ML cost function is accurate. It corresponds to the object shape that better describes the entire set of multiple observed images.

Inference of 3D rigid structure: surface-based rank 1 factorization

We develop algorithms that recover 3D structure from a monocular video sequence by processing a sequence of multiple frames, rather than only two. We avoid the quagmire of tracking a large set of features by describing the 3D shape of the scene by a parameterization of its surface. This parametric description of the 3D surface induces, in turn, a parametric model for the 2D motion of the brightness pattern in the image plane. We start by tracking regions where the 2D motion in the image plane is described by a single set of parameters. Then, our method estimates the parameters that describe the 3D shape

and the 3D motion by factorizing a matrix that collects the parameters describing the 2D motion in the image plane. We show that this matrix is rank 1 in a noiseless situation; our method finds suitable factors of this matrix by using a fast algorithm to compute its largest singular value and associated singular vectors.

According to the analogy of Figure 1.2, the problem of recovering 3D structure from a 2D video sequence is formulated as the joint ML estimation of all the unknowns: the 3D motions, the 3D shape of the object, and the texture of the object. Again, we do not attempt the direct minimization of the ML cost function with respect to the entire set of parameters by using generic optimization methods. Rather, we exploit the specific characteristics of the problem to develop a computationally feasible approximation to the ML solution. We start by expressing the estimate of the texture in terms of the 3D shape and 3D motion parameters. Then, we replace the texture estimate into the ML cost function and are left with a cost function that depends on the 3D shape and 3D motion parameters only through the 2D motion of the brightness pattern in the image plane. This way, we show that the classic *structure from motion* (SFM) approach is in fact an approximation to the ML estimate of the 3D structure.

To compute the 3D structure in an expedite way, we develop the *surface-based rank 1 factorization method*. We parameterize the 3D shape of the object surface. This parameterization induces a parametric description for the 2D motion in the image plane. We start by computing the parameters describing the 2D motion in the image plane for a set of regions. This step corresponds to the minimization of a simplified version of the ML cost function with respect to the 2D motion parameters. The relation between the parameters that describe the 3D structure and the parameters describing the 2D motion depends on the geometric projection mechanism. We use the orthographic projection model that is known to be a good approximation to the perspective projection when the relative depth of the scene is small when compared to the distance to the camera. The relation between the two sets of parameters referred above enables us to recover the parameters describing the 3D structure by factorizing a matrix that collects the 2D motion parameters. We

show that this matrix is rank 1 in a noiseless situation. The estimates of the 3D motion parameters and the 3D shape parameters are then obtained from the column and row vectors whose product best matches the data in the matrix of 2D motion parameters. We factorize this matrix by using a fast algorithm to compute its largest singular value and the associated singular vectors.

Our technique is simultaneously a generalization of the original feature-based factorization method of Tomasi and Kanade [59, 61], and a further step into the use of linear subspace constraints in motion analysis. The surface-based rank 1 factorization is a generalization of the original factorization method because the original feature-based approach of [59, 61] corresponds to a special case of our framework where the 3D shape is described by a set of pointwise patches with constant depth. The surface-based rank 1 factorization method is also a further step into the use of subspace constraints in motion analysis because we show how to reduce the dimension of the space of the measurement matrix by making an adequate linear subspace projection. This projection enables us to recover SFM by factorizing a matrix that is rank 1 in a noiseless situation, rather than a rank 3 matrix like in the original factorization method. This simplifies both the decomposition and normalization steps involved in the factorization approaches.

Since the accuracy of the 2D motion estimates is not the same for all image regions – it depends on the the size of the estimating region and on the variability of the image intensity pattern, – a robust estimate of the 3D structure would weight more the contribution of more accurate estimates of 2D motion. To compute this weighted estimate in an expedite way, we develop the *weighted factorization method* that takes into account the accuracy of the 2D motion estimates without paying additional computational cost. This is achieved by rewriting the problem with weights as the non-weighted factorization of a modified matrix.

The estimate of the 3D shape provided by the SFM step is described by the set of depths of the regions for which the 2D motion was computed. This estimate of the 3D shape can be a rough approximation of the true 3D shape because the set of regions

for which the 2D motion was computed can be sparse. To overcome this fact, we propose a final step that estimates the 3D shape directly from the image intensity values. We introduce the 3D motion estimates into the ML cost function and minimize it with respect to the 3D shape. To accomplish this minimization, we use a computationally simple *continuation-type multiresolution* algorithm. Our algorithm starts by estimating coarse approximations to the 3D shape. Then, it refines the estimate as more images are being taken into account. The computational simplicity of our algorithm comes from the fact that each stage, although non-linear, is solved by a simple *Gauss-Newton method* that requires no more than two or three iterations.

Other researchers also estimate the 3D structure directly from the image intensity values. Horn and Weldon [29] estimate the 3D structure parameters by using the *brightness change constraint* between two consecutive frames. Heel [28] builds on this work by using a *Kalman filter* to update the estimates over time. As outlined in the paragraph above, we infer 3D structure directly from the image intensity values, through ML estimation, but, in contrast with [28], we model the rigidity of the scene over the set of available frames, rather than trying to fuse a set of possibly inaccurate estimates obtained from pairs of consecutive frames.

1.3 Thesis Overview

The thesis is organized in two parts. Part I addresses the problem of segmenting a two-dimensional (2D) rigid moving object. Part II addresses the problem of recovering three-dimensional (3D) rigid structure from a monocular video sequence. Part I includes chapters 2 through 6, while Part II corresponds to chapters 7 through 11. Chapter 12 concludes the thesis. The following paragraphs overview the contents of each chapter of the thesis.

Part I

Chapter 2 introduces the 2D motion segmentation problem. We identify the limitations of current methods, in particular when the object is low textured or moves against low contrast background. Then, we describe the observation model and formulate the *Maximum Likelihood* (ML) estimate. The 2D shape of the moving object is described by a discretized binary template. The high flexibility of this description enables the accurate representation of non-smooth shapes, as for example the shape of a boat with a mast (such as the one appearing in the coast-guard test sequence [40] for the MPEG-4 video standard [57]), or complex shapes, as for example the shape of objects with holes. Each pixel of each frame in the video sequence is modeled as a noisy version either of the moving object intensity level (object texture), if that pixel belongs to the template of the moving object, or of the background intensity level (background texture), if otherwise. The object moves relative to the background, covering some regions of the background and uncovering other regions. Also the position of the background relative to the camera changes, due to the camera motion, letting different views of the background appear in the camera window. The segmentation problem is formulated as the ML estimation of all the unknowns from the video sequence: the motions of the object, the motions of the camera, the binary template of the object, the texture of the object, and the texture of the background. This chapter also introduces important notation used in the thesis.

To minimize the ML cost function, we decouple the estimation of the motions from the estimation of the remaining parameters. The motions are estimated on a frame by frame basis. Chapter 3 is devoted to the estimation of the motion of the brightness pattern in the image plane. We show that, for a single pair of frames, the ML estimation leads to the usual approach of minimizing the brightness difference between those two frames. Then, we use known methods to estimate the motion of the brightness pattern for two well known 2D motion models: the translational motion model and the affine motion model. An important contribution in this chapter is the derivation of expressions for the variance

of the error of the estimates of the motion parameters in terms of the spatial variability of the brightness pattern and the size of the region of analysis. These expressions provide an easy way to predict the expected error in estimating the 2D motion of the brightness pattern for a given region of the image. Basically, the larger the region or the larger the variability of the brightness pattern, the more accurate are the estimates of the 2D motion parameters.

Chapter 4 describes our two-step iterative algorithm to segment the 2D rigid moving object. To minimize the ML cost function, we start by estimating the texture of the moving object. By minimizing the ML cost function with respect to the object texture, we obtain a closed-form expression. We insert the estimate of the object texture into the ML cost function and are left with a function of the texture of the background and the template of the moving object. Our algorithm minimizes this cost function by estimating iteratively the following: (i) the background texture with known template; and (ii) the object template with known background. The background estimate in step (i) is obtained in closed-form. The template estimate in step (ii) leads to a simple binary test evaluated at each pixel. We illustrate our approach with an experiment where we used real video frames to construct a challenging sequence where an object with complex template moves against low contrast background.

Chapter 5 analyzes the behavior of the segmentation algorithm. We present a statistical analysis of the binary test involved in the two-step iterative algorithm and illustrate with one experiment with synthetic data the behavior of the test.

Chapter 6 describes experiments with real video sequences that illustrate the performance of the segmentation algorithm.

Part II

Chapter 7, the first chapter of Part II, states the problem of inferring 3D structure from an image sequence. After reviewing the bibliography and contrasting our approach with related work, we describe the observation model and formulate the ML solution for this

problem. We consider one single object. The frames in the video sequence are modeled as the orthogonal projection of the texture of the object plus noise. The orthogonal projection is known to be a good approximation to the perspective projection when the relative depth of the scene is small when compared to the distance to the camera (in this scenario, it is not possible to recover the 3D shape by inferring the absolute depth). The mapping of the texture of the object to the image plane depends both on the 3D shape of the object and the 3D position of the object relative to the camera. The problem of inferring 3D structure is then stated as the estimation of the 3D shape of the object, the 3D motion of the object relative to the camera, and the texture of the object from the video sequence. By minimizing the ML cost function with respect to the object texture, we express the ML estimate of the texture in terms of the remaining unknowns. We replace the texture estimate into the ML cost function, obtaining a cost function that depends on the 3D shape and 3D motion through the 2D motion induced in the image plane. This way, we show that the usual *structure from motion* (SFM) approach can be seen as an approximation to the ML solution – when inferring SFM, we minimize the ML cost function by first computing the 2D motion of the brightness pattern in the image plane, then estimating the 3D shape and 3D motion from the 2D motion.

Chapters 8 and 9 are devoted to SFM. They detail our approach to the recovery of 3D motion and 3D shape from the 2D motion of the brightness pattern of a sequence of images of a rigid scene. In chapter 8 we introduce the *rank 1 factorization*. We consider the scenario used by Tomasi and Kanade to introduce the *factorization method* [59, 61]. A set of features is tracked across a set of frames. The 3D shape is represented by the set of 3D positions of the feature points. In the formulation of Tomasi and Kanade, the 3D shape and 3D motion are computed by using *Singular Value Decomposition* (SVD) to approximate a measurement matrix that is rank 3 in a noiseless situation. We reformulate the problem using the fact that the feature coordinates along the image plane are known from their projection in the first frame. By using an appropriate linear subspace projection, we recover the 3D shape, i.e., the relative depths of the features, and the 3D motion,

from the set of feature trajectories, by a simple factorization of a matrix that is rank 1 in a noiseless situation. This leads to a very fast algorithm to recover 3D structure from 2D motion, even when using a large number of features and a large number of frames. This chapter includes an experimental comparison between the computational cost of the rank 1 factorization and the one of the original factorization method of Tomasi and Kanade [59, 61].

When the goal is the recovery of a dense representation of the 3D shape, the feature-based approach may not solve the problem satisfactorily because it may require tracking a very large number of features to obtain a dense description of the 3D shape. This leads to a complex, if not impossible, correspondence problem because only distinguished points, as brightness corners, can be accurately tracked. To overcome this difficulty, we introduce in chapter 9 the *surface-based factorization method*. We represent the 3D shape by a parametric description of the object surface. This representation induces a parametric model for the 2D motion of the brightness pattern in the image plane. For example, for piecewise planar 3D shapes, the 2D motion in the image plane is described by the affine motion model with different parameterizations for regions corresponding to different surface patches. We start by estimating the parameters describing the 2D motion by using the method described in chapter 3. Then, we recover the parameters that describe the 3D shape and the 3D motion from the parameters describing the 2D motion in the image plane. This is done by factorizing a surface-based measurement matrix that collects the set of 2D motion parameters. To factorize this matrix in an efficient way, we use the methodology of the rank 1 factorization of chapter 8. In chapter 9 we also show how to include confidence weights in the estimates of the 2D motion parameters when recovering the 3D structure, i.e., how to weight more the contribution of a large region than the contribution of a small region, or the contribution of a region with a “highly contrasted” brightness pattern than the contribution of a region with “smooth” brightness pattern. These weights are computed from the variances of the estimates of the 2D motion parameters, as detailed in chapter 3. We introduce the *weighted factorization* to recover

the 3D structure using the confidence weights without additional computational cost. This is accomplished by rewriting the problem as the factorization of a modified measurement matrix, and using the rank 1 factorization method to factorize this matrix.

In chapters 8 and 9 we estimate the 3D shape and the 3D motion from the 2D motion of the brightness pattern in the image plane. As mentioned above, this procedure approximates the ML estimation of the 3D structure. The quality of the estimate of the 3D shape obtained this way is limited by the possibility of estimating the 2D motion over the entire image. Since for textures that exhibit a predominant spatial orientation it is very difficult to compute the 2D motion of the brightness pattern (this problem is studied in detail in chapter 3), the estimate of the 3D shape obtained by inferring SFM may be much rougher than the corresponding ML estimate. In chapter 10 we propose a method to estimate the 3D shape directly from the image intensity values by minimizing the ML cost function, after introducing the estimates of the 3D motions. Our approach provides an efficient way to cope with the ill-posedness of estimating the motion in the image plane. In fact, the local *brightness change constraint* leads to a single restriction, which is insufficient to determine the two components of the local image motion (the so called *aperture problem*). Our method of estimating directly the 3D shape overcomes the aperture problem because we are left with the local depth as a single unknown, after computing the 3D motion in the first step. To minimize the ML cost function we develop a *multiresolution continuation-type* algorithm that works by estimating coarse approximations to the 3D shape at the beginning and refining the estimate as more images are being taken into account. Each stage of the continuation algorithm uses a *Gauss-Newton method* to update the estimate of the 3D shape. The derivatives involved in the Gauss-Newton method are obtained from image gradients in a very simple way.

Chapters 8, 9, and 10 include experiments using synthetic data to illustrate the ideas, methodologies, and algorithms proposed. In chapter 11 we describe experiments that demonstrate the performance of our algorithms in recovering 3D structure from real video sequences.

The thesis ends with chapter 12 where we summarize our work, emphasize the original contributions, and point out possible future research directions.

Parts of the work described in the thesis are published in references [1-9].

Part I

Segmentation of 2D Moving Object

Chapter 2

Statement of the Segmentation Problem

2.1 Introduction

Motion segmentation methods often fail to detect the motions of low textured regions. We develop an algorithm for segmentation of low textured moving objects. While usually current motion segmentation methods use only two or three consecutive images our method refines the shape of the moving object by processing successively the new frames as they become available. By integrating across time the information content of the image sequence, we achieve accurate motion-based segmentation for general unstructured scenes.

The segmentation of an image into regions that undergo different motions has received the attention of a large number of researchers. According to their research focus, different scientific communities addressed the motion segmentation task from distinct viewpoints. In section 2.2, we overview the common approaches in the video coding and computer vision communities and relate ours to previously published research work.

We formulate image sequence analysis as a parameter estimation problem. As introduced in chapter 1, we use the analogy between a communications system and image sequence analysis. The segmentation algorithm is derived as a computationally simple approximation to the *Maximum Likelihood* (ML) estimate of the parameters involved in

the two-dimensional (2D) image sequence model: the motions, the template of the moving object, its intensity levels (the object texture), and the intensity levels of the background pixels (the background texture). The joint ML estimation of the complete set of parameters is a very complex task. Motivated by our experience with real video sequences, we decouple the estimation of the motions (moving objects and camera) from that of the remaining parameters. The motions are estimated on a frame by frame basis and then used in the estimation of the remaining parameters. Then, we introduce the motion estimates into the ML cost function and minimize this function with respect to the remaining parameters.

The estimate of the object texture is obtained in closed form. To estimate the background texture and the moving object template, we develop a fast two-step iterative algorithm. The first step estimates the background for a fixed template – the solution is obtained in closed form. The second step estimates the template for a fixed background – the solution is given by a simple binary test evaluated at each pixel. The algorithm converges in a few iterations, typically three to five iterations.

This chapter states the 2D rigid object segmentation problem. Section 2.3 introduces notation. In section 2.4 we formulate the segmentation problem according to the analogy between a communications system and image sequence analysis, as introduced in chapter 1. We detail the observation model and the unknown parameters involved. Section 2.5 derives the ML estimate and outlines our approach to the minimization of the ML cost function. Section 2.6 summarizes the content of the chapter.

2.2 Related Work

Several papers on image sequence coding address the motion segmentation task with computation time concerns. They reduce temporal redundancy by predicting each frame from the previous one through motion compensation. See reference [39] for a review on very low bit rate video coding. Regions undergoing different movements are compensated in differ-

ent ways, according to their motion. This type of approach is known as implicit model-based image sequence coding. *Implicit modeling* means that the three-dimensional (3D) structure of the scene is taken into account implicitly by a two-dimensional (2D) model representing its projection onto the image plane. This is in opposition to *explicit modeling*, where the 3D structure is taken into account explicitly.

The techniques used in image sequence coding attempt to segment the moving objects by processing only two consecutive frames. Since their focus is on compression and not in developing a high level representation, these efforts have not considered low textured scenes, and regions with no texture are considered unchanged. As an example, we applied the algorithm of reference [23] to segmenting a low textured moving object. Two consecutive frames of a traffic road video clip are shown in the left side of Figure 1.1, see section 1.1. In the right side of Figure 1.1, the template of the moving car was found by excluding from the regions that changed between the two co-registered frames the ones that correspond to uncovered background areas, see reference [23]. The small regions that due to the noise are misclassified as belonging to the car template can be discarded by an adequate morphological post-processing. However, due to the low texture of the car, the regions in the interior of the car are misclassified as belonging to the background, leading to a highly incomplete car template.

High level representation in image sequence understanding has been considered in the computer vision literature. Their approach to motion-based segmentation copes with low textured scenes by coupling motion-based segmentation with prior knowledge about the scenes as in statistical regularization techniques, or by combining motion with other attributes. For example, reference [25] uses a *Markov Random Field* (MRF) prior and a *Bayesian Maximum a Posteriori* (MAP) criterion to segment moving regions. The authors suggest a multiscale MRF modeling to resolve large regions of uniform intensity. In reference [16], the contour of a moving object is estimated by fusing motion with color segmentation and edge detection. In general, these methods lead to complex and time consuming algorithms.

References [31, 32] describe one of the few approaches using temporal integration by averaging the images registered according to the motion of the different objects in the scene. After processing a number of frames, each of these integrated images is expected to show only one sharp region corresponding to the tracked object. This region is found by detecting the stationary regions between the corresponding integrated image and the current frame. Unless the background is textured enough to blur completely the averaged images, some regions of the background can be classified as stationary. In this situation, their method overestimates the template of the moving object. This is particularly likely to happen when the background has large regions with almost constant color or intensity level.

Our approach is related to the approach of references [31, 32], however, we model explicitly the occlusion of the background by the moving object and we use all the frames available rather than just a single frame to estimate the moving object template. Even when the moving object has a color very similar to the color of the background, our algorithm has the ability to resolve accurately the moving object from the background, because it integrates over time those small differences.

2.3 Notation

We discuss motion segmentation in the context of *Generative Video* (GV), see references [34, 35, 36, 37]. GV is a framework for the analysis and synthesis of video sequences. In GV the operational units are not the individual images in the original sequence, as in standard methods, but rather the world images and the ancillary data. The world images encode the non-redundant information about the video sequence. They are augmented views of the world – background world image – and complete views of moving objects – figure world images. The ancillary data registers the world images, stratifies them at each time instant, and positions the camera with respect to the layering of world images. The world images and the ancillary data are the GV representation, the information that

is needed to regenerate the original video sequence. We formulate the moving object segmentation task as the problem of generating the world images and ancillary data for the GV representation of a video clip.

We use the analogy between a communications system and an image analysis task, as introduced in chapter 1 and illustrated in Figure 1.2. According to this approach, a key step is the specification of a model for the video sequence. To introduce this model, we first define building blocks such as image, world image, template, and registration.

An image is a real function defined on a subset of the real plane. The image space is a set $\{\mathbf{I} : \mathcal{D} \rightarrow \mathcal{R}\}$, where \mathbf{I} is an image, \mathcal{D} is the domain of the image, and \mathcal{R} is the range of the image. The domain \mathcal{D} is a compact subset of the real plane \mathbb{R}^2 , and the range \mathcal{R} is a subset of the real line \mathbb{R} . Examples of images are the frame f in the video sequence, denoted by \mathbf{I}_f , the background world image, denoted by \mathbf{B} , the moving object world image, denoted by \mathbf{O} , and the moving object template, denoted by \mathbf{T} . The images \mathbf{I}_f , \mathbf{B} , and \mathbf{O} have range $\mathcal{R} = \mathbb{R}$. They code intensity gray levels¹. The template of the moving object is a binary image, i.e., an image with range $\mathcal{R} = \{0, 1\}$, defining the region occupied by the moving object. The domain of the images \mathbf{I}_f and \mathbf{T} is a rectangle corresponding to the support of the frames. The domain of the background world image \mathbf{B} is a subset \mathcal{D} of the plane whose shape and size depends on the camera motion, i.e., \mathcal{D} is the region of the background observed in the entire sequence. The domain \mathcal{D} of the moving object world image is the subset of \mathbb{R}^2 where the template \mathbf{T} takes the value 1, i.e., $\mathcal{D} = \{(x, y) : \mathbf{T}(x, y) = 1\}$.

In our implementation, the domain of each image is rectangular shaped with size fitting the needs of the corresponding image. Although we use a continuous spatial dependence for commodity, in practice the domains are discretized and the images are stored as

¹The intensity values of the images in the video sequence are positive. In our experiments, these values are coded by a binary word of eight bits. Thus, the intensity values of a gray level image are in the set of integers in the interval $[0, 255]$. For simplicity, we do not take into account the discretization and the saturations, i.e., we consider the intensity values to be real numbers and the gray level images to have range $\mathcal{R} = \mathbb{R}$. The analysis in the thesis is easily extended to color images. A color is represented by specifying three intensities, either of the perceptual attributes *brightness*, *hue*, and *saturation*; or of the primary colors *red*, *green*, and *blue*, see reference [33]. The range of a color image is then $\mathcal{R} = \mathbb{R}^3$.

matrices. We index the entries of each of these matrices by the pixels (x, y) of each image and refer to the value of image \mathbf{I} at pixel (x, y) as $\mathbf{I}(x, y)$. Throughout the text, we refer to the image product of two images \mathbf{A} and \mathbf{B} , i.e., the image whose value at pixel (x, y) equals $\mathbf{A}(x, y)\mathbf{B}(x, y)$, as the image \mathbf{AB} . Note that this product corresponds to the Hadamard product, or elementwise product, of the matrices representing images \mathbf{A} and \mathbf{B} , not their matrix product.

We consider two-dimensional (2D) parallel motions, i.e., all motions (translations and rotations) are parallel to the camera plane. We represent this kind of motions by specifying time varying position vectors. These vectors code rotation-translation pairs that take values in the group of rigid transformations of the plane, the special Euclidean group $\text{SE}(2)$. The image obtained by applying the rigid motion coded by the vector \mathbf{p} to the image \mathbf{I} is denoted by $\mathcal{M}(\mathbf{p})\mathbf{I}$. The image $\mathcal{M}(\mathbf{p})\mathbf{I}$ is also usually called the registration of the image \mathbf{I} according to the position vector \mathbf{p} . The entity represented by $\mathcal{M}(\mathbf{p})$ is seen as a motion operator. In practice, the (x, y) entry of the matrix representing the image $\mathcal{M}(\mathbf{p})\mathbf{I}$ is given by $\mathcal{M}(\mathbf{p})\mathbf{I}(x, y) = \mathbf{I}(f_x(\mathbf{p}; x, y), f_y(\mathbf{p}; x, y))$ where $f_x(\mathbf{p}; x, y)$ and $f_y(\mathbf{p}; x, y)$ represent the coordinate transformation imposed by the 2D rigid motion. We use bilinear interpolation to compute the intensity values at points that fall in between the stored samples of an image.

The motion operators can be composed. The registration of the image $\mathcal{M}(\mathbf{p})\mathbf{I}$ according to the position vector \mathbf{q} is denoted by $\mathcal{M}(\mathbf{qp})\mathbf{I}$. By doing this we are using the notation \mathbf{qp} for the composition of the two elements of $\text{SE}(2)$, \mathbf{q} and \mathbf{p} . We denote the inverse of \mathbf{p} by $\mathbf{p}^\#$, i.e., the vector $\mathbf{p}^\#$ is such that when composed with \mathbf{p} we obtain the identity element of $\text{SE}(2)$. Thus, the registration of the image $\mathcal{M}(\mathbf{p})\mathbf{I}$ according to the position vector $\mathbf{p}^\#$ obtains the original image \mathbf{I} , so we have $\mathcal{M}(\mathbf{p}^\#\mathbf{p})\mathbf{I} = \mathcal{M}(\mathbf{pp}^\#)\mathbf{I} = \mathbf{I}$. Note that, in general, the elements of $\text{SE}(2)$ do not commute, i.e., we have $\mathbf{qp} \neq \mathbf{pq}$, and $\mathcal{M}(\mathbf{qp})\mathbf{I} \neq \mathcal{M}(\mathbf{pq})\mathbf{I}$. Only in special cases is the composition of the motion operators not affected by the order of application, as for example when the motions \mathbf{p} and \mathbf{q} are pure translations or pure rotations.

The notation for the position vectors involved in the segmentation problem is as follows. The vector \mathbf{p}_f represents the position of the background world image relative to the camera in frame f . The vector \mathbf{q}_f represents the position of the moving object relative to the camera in frame f .

2.4 Problem Formulation

Following the analogy between a communications system and an image analysis task, as introduced in chapter 1 and illustrated in Figure 1.2, we specify the observation model describing the video sequence.

The observation model considers a scene with a moving object in front of a moving camera with two-dimensional (2D) parallel motions. The pixel (x, y) of the image \mathbf{I}_f belongs either to the background world image \mathbf{B} or to the object world image \mathbf{O} . The intensity $\mathbf{I}_f(x, y)$ of the pixel (x, y) is modeled as

$$\mathbf{I}_f(x, y) = \mathcal{M}(\mathbf{p}_f^\#)\mathbf{B}(x, y) \left[1 - \mathcal{M}(\mathbf{q}_f^\#)\mathbf{T}(x, y) \right] + \mathcal{M}(\mathbf{q}_f^\#)\mathbf{O}(x, y) + \mathbf{W}_f(x, y). \quad (2.1)$$

In equation (2.1), \mathbf{T} is the moving object template, \mathbf{p}_f and \mathbf{q}_f are the camera pose and the object position, and \mathbf{W}_f stands for the observation noise, assumed Gaussian, zero mean, and white.

Equation (2.1) states that the intensity of the pixel (x, y) on frame f , $\mathbf{I}_f(x, y)$, is a noisy version of the true value of the intensity level of the pixel (x, y) . If the pixel (x, y) of the current image belongs to the template of the object, \mathbf{T} , after the template is compensated by the object position, i.e., registered according to the vector $\mathbf{q}_f^\#$, then $\mathcal{M}(\mathbf{q}_f^\#)\mathbf{T}(x, y) = 1$. In this case, the first term of the right hand side of (2.1) is zero, while the second term equals $\mathcal{M}(\mathbf{q}_f^\#)\mathbf{O}(x, y)$, the intensity of the pixel (x, y) of the moving object. In other words, the intensity $\mathbf{I}_f(x, y)$ equals the object intensity $\mathcal{M}(\mathbf{q}_f^\#)\mathbf{O}(x, y)$ corrupted by the noise $\mathbf{W}_f(x, y)$. On the other hand, if the pixel (x, y) does not belong to the template of the object, $\mathcal{M}(\mathbf{q}_f^\#)\mathbf{T}(x, y) = 0$, and this pixel belongs to the background

world image \mathbf{B} , registered according to the inverse $\mathbf{p}_f^\#$ of the camera position. In this case, the intensity $\mathbf{I}_f(x, y)$ is a noisy version of the background intensity $\mathcal{M}(\mathbf{p}_f^\#)\mathbf{B}(x, y)$. We want to emphasize that rather than modeling simply the two different motions, as usually done when processing only two consecutive frames, expression (2.1) models the occlusion of the background by the moving object explicitly.

Expression (2.1) is rewritten in compact form as

$$\mathbf{I}_f = \left\{ \mathcal{M}(\mathbf{p}_f^\#)\mathbf{B} \left[\mathbf{1} - \mathcal{M}(\mathbf{q}_f^\#)\mathbf{T} \right] + \mathcal{M}(\mathbf{q}_f^\#)\mathbf{O} \mathcal{M}(\mathbf{q}_f^\#)\mathbf{T} + \mathbf{W}_f \right\} \mathbf{H}, \quad (2.2)$$

where we assume that $\mathbf{I}_f(x, y) = 0$ for (x, y) outside the region observed by the camera. This is taken care of in equation (2.2) by the binary image \mathbf{H} whose (x, y) entry is such that $\mathbf{H}(x, y) = 1$ if pixel (x, y) is in the observed images \mathbf{I}_f or $\mathbf{H}(x, y) = 0$ if otherwise. The image $\mathbf{1}$ is constant with value 1.

Figure 2.1 illustrates expression (2.2) for one-dimensional (1D) frames. The top plot, a sinusoid, is the background world image \mathbf{B} . The template \mathbf{T} of the moving object is the union of the two disjoint intervals shown on the left of the second level. The intensity level of the moving object \mathbf{O} is also sinusoidal, see the right plot on the second level. Its frequency is higher than the background frequency. The camera window \mathbf{H} is the interval shown in the third level. It clips the region observed by the camera. Two frames \mathbf{I}_1 and \mathbf{I}_2 are shown in the two bottom curves. They are given by a noise-free version of the model in expression (2.2). In between these two frames, both the camera and the object moved: the camera moved 2 pixels to the right, corresponding to the background motion in the opposite direction, while the object moved 3 pixels to the right relative to the camera. The observation model of expression (2.2) and the illustration of Figure 2.1 emphasize the role of the building blocks involved in representing an image sequence $\{\mathbf{I}_f, 1 \leq f \leq F\}$ according to the *Generative Video* (GV) framework [34, 35, 36, 37].

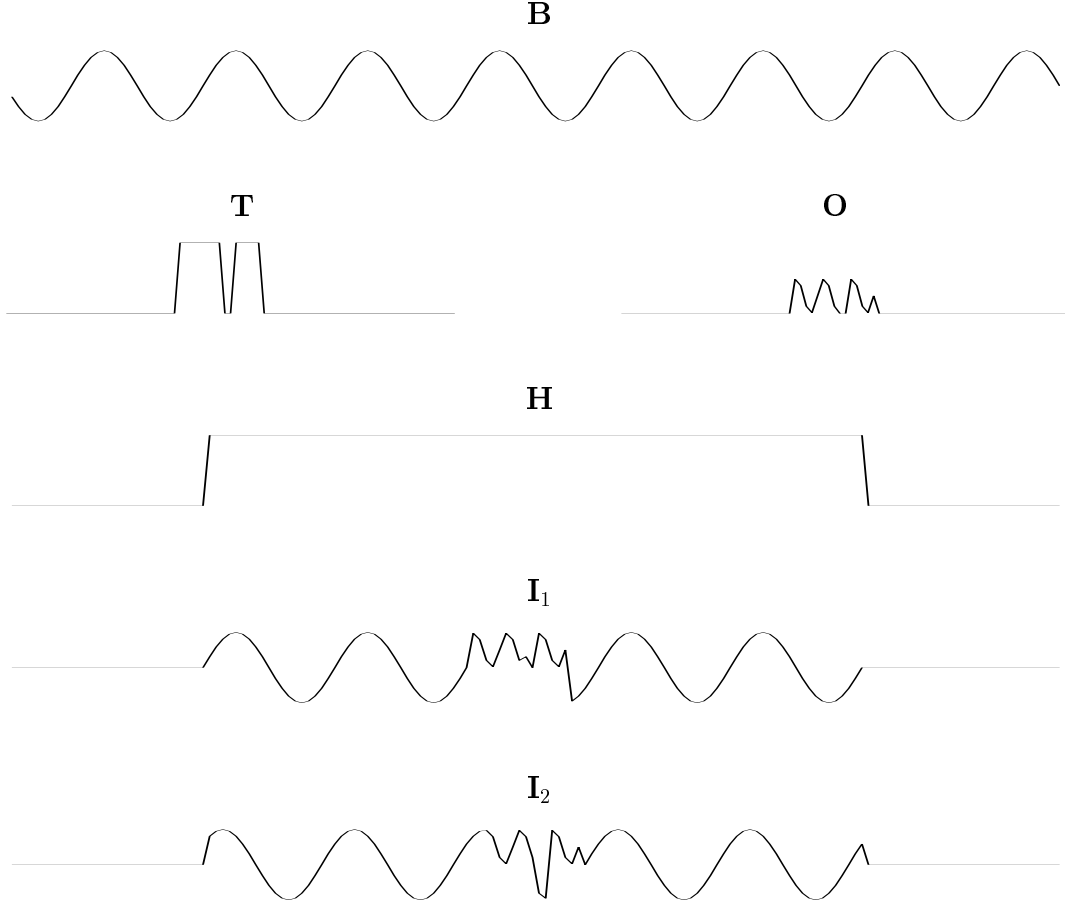


Figure 2.1: Illustration of the 1D GV image formation and observation model.

Multiframe segmentation from motion

We formulate the segmentation problem as the recovery of the quantities that define the GV representation of the video sequence. Given F frames $\{\mathbf{I}_f, 1 \leq f \leq F\}$, we want to estimate the background world image \mathbf{B} , the object world image \mathbf{O} , the object template \mathbf{T} , the camera poses $\{\mathbf{p}_f, 1 \leq f \leq F\}$, and the object positions $\{\mathbf{q}_f, 1 \leq f \leq F\}$. The quantities $\{\mathbf{B}, \mathbf{O}, \mathbf{T}, \{\mathbf{p}_f, 1 \leq f \leq F\}, \{\mathbf{q}_f, 1 \leq f \leq F\}\}$ define the GV representation, the information that is needed to regenerate the original video sequence.

2.5 Maximum Likelihood Estimation

Using the observation model of expression (2.2) and the Gaussian white noise assumption, *Maximum Likelihood* (ML) estimation leads to the minimization over all *Generative Video* (GV) [34, 35, 36, 37] parameters of the functional²

$$C_2 = \iint \sum_{f=1}^F \left\{ \mathbf{I}_f(x, y) - \mathcal{M}(\mathbf{p}_f^\#) \mathbf{B}(x, y) \left[1 - \mathcal{M}(\mathbf{q}_f^\#) \mathbf{T}(x, y) \right] - \mathcal{M}(\mathbf{q}_f^\#) \mathbf{O}(x, y) \mathcal{M}(\mathbf{q}_f^\#) \mathbf{T}(x, y) \right\}^2 \mathbf{H}(x, y) dx dy, \quad (2.3)$$

where the inner sum is over the full set of F frames and the outer integral is over all pixels.

The estimation of the parameters of expression (2.2) using the F frames rather than a single pair of images is a distinguishing feature of our work. Other techniques usually process only two or three consecutive frames. We use all frames available as needed. The estimation of the parameters through the minimization of a cost function that involves directly the image intensity values is another distinguishing feature of our approach. Other methods try to make some type of post-processing over incomplete template estimates. We process directly the image intensity values, through ML estimation.

The minimization of the functional C_2 in equation (2.3) with respect to the set of GV constructs $\{\mathbf{B}, \mathbf{O}, \mathbf{T}\}$ and to the motions $\{\{\mathbf{p}_f\}, \{\mathbf{q}_f\}, 1 \leq f \leq F\}$ is a highly complex task. To obtain a computationally feasible algorithm, we simplify the problem. We decouple the estimation of the motions $\{\{\mathbf{p}_f\}, \{\mathbf{q}_f\}, 1 \leq f \leq F\}$ from the determination of the GV constructs $\{\mathbf{B}, \mathbf{O}, \mathbf{T}\}$. This is reasonable from a practical point of view and is well supported by our experimental results with real videos.

The rationale behind the simplification is that the motion of the object (and the motion of the background) can be inferred without having the knowledge of the exact object template. When only two or three frames are given, even humans find it much

²We use a continuous spatial dependence for commodity. The variables x and y are continuous while f is discrete. In practice, the integral is approximated by the sum over all the pixels.

easier to infer the motions present in the scene than to recover an accurate template of the moving object. To better appreciate the complexity of the problem, the reader can imagine an image sequence for which there is not prior knowledge available, except that there is a background and an occluding object that moves differently from the background. Since there are no spatial cues, consider, for example, that the background texture and the object texture are spatial white noise random variables. In this situation, humans can easily infer the motion of the background and the motion of the object, even from only two consecutive frames. With respect to the template of the moving object, we are able to infer much more accurate templates if we are given a higher number of frames because in this case we easily capture the rigidity of the object across time. This observation motivated our approach of decoupling the estimation of the motions from the estimation of the remaining parameters.

We perform the estimation of the motions on a frame by frame basis by minimizing a simplified version of the ML cost function. Chapter 3 studies in detail the motion estimation method. After estimating the motions, we introduce the motion estimates into the ML cost function and minimize with respect to the remaining parameters. The minimization procedure is described in chapter 4. It uses a two-step iterative algorithm that determines, recursively, the estimate of the background and the estimate of the moving object template.

The solution provided by our algorithm is sub-optimal, in the sense that it is an approximation to the ML estimate of the entire set of parameters. As discussed in chapter 1, the solution provided by our algorithm can be seen as an initial guess for the minimizer of the ML cost function given by expression (2.3). Then, we can refine the estimate by using a greedy approach. We must restate, however, that the key problem here is to find the initial guess in an expedite way, not the final refinement.

2.6 Summary

This chapter introduces the problem of segmenting a two-dimensional (2D) rigid moving object from a video sequence. We motivate the problem by illustrating the limitations of the two-frame based algorithms. In contrast, our approach models the rigidity of the object over a larger set of frames.

We formulate the 2D segmentation problem by using the analogy between a communications system and an image analysis task. We introduce the observation model, that is based on the *Generative Video* (GV) [34, 35, 36, 37] image formation model. Rather than modeling only the different motions, our model also makes explicit the occlusion of the background by the moving object.

The segmentation algorithm we propose works directly with the image intensity values, through *Maximum Likelihood* (ML) estimation, rather than trying to make some kind of post-processing over template estimates. We start by formulating the ML estimate of the entire set of unknowns involved, leading to the minimization of a complex cost function. Then, we motivate and outline our approach to the minimization of the ML cost function.

Chapter 3

Image Motion Estimation

3.1 Introduction

This chapter is about the estimation of the motion of the brightness pattern in the image plane. This is a crucial step in solving the problems addressed in this thesis and, in general, in any motion analysis task. Both the problem of segmenting a two-dimensional (2D) rigid object and the problem of inferring three-dimensional (3D) rigid structure require the step of estimating 2D motion to accomplish their higher level goals. The problem of estimating the motion of the brightness pattern has been widely addressed in the recent past. Any known numerical technique to estimate the motion can be used in our approach. To make the thesis self-contained, we overview the estimation method that we use. The most significant original contribution of this chapter is the study of the estimation error. We derive expressions that approximate the variance of the estimation error. The variance of the estimation error is a measure of the accuracy of the 2D motion estimate. We use this measure of accuracy to weight differently the 2D motion estimates when inferring 3D structure from 2D motion, as done in Part II, chapter 9, section 9.4.

We estimate the motions on a frame by frame basis. The cue to estimate the motion of the brightness pattern between two frames is the brightness constancy. Since the early days of motion analysis, researchers have noticed that local motion estimation is an ill-posed problem, see for example reference [30]. In fact, it is easily shown that it is not possible to determine the 2D motion if we are allowed to see only through an infinitesimally

small spatial aperture. This limitation is usually referred to as the *aperture problem*. To overcome the aperture problem, smoothing assumptions are commonly made. Several smoothing techniques have been reported in the literature. In our work, we compute the position vectors by fitting parametric motion models to regions of the image. To estimate the motion parameters we use an approach made popular by Bergen, Anandan, Hanna, and Hingorani, in reference [13]. This approach uses a hierarchical *Gauss-Newton method* where the derivatives involved are computed from the image gradients.

The motion of the brightness pattern of the scenes we are interested in is not described by a unique parametric model for the entire image. When segmenting the 2D rigid moving object, we will have two different parameterizations, one for the region corresponding to the background, the other for the region corresponding to the moving object. When inferring 3D rigid structure, we will have different parameterizations for regions corresponding to different parameterizations of the 3D shape. This leads us to the more complex problem of estimating simultaneously the support regions and the motion parameters. This problem was also well addressed in the past, see for example references [11, 18, 70]. Reference [11] suggests a classification of the methods that estimate simultaneously the support regions and motion parameters into two classes: the *sequential methods* and the *competitive methods*. The sequential methods start by estimating the motion parameters that best describe the motion of the entire image. Then, the images are co-registered according to the estimated motion. The pixels where the registered frame difference is below a threshold are considered to belong to the dominant region. Finally, the dominant region is discarded and the process is repeated with the remaining pixels. The competitive methods use greedy algorithms to optimize, iteratively, criteria that take into account both the fitting accuracy and the model complexity. These algorithms are in general computationally heavy.

Since we do not require an accurate segmentation when estimating the image motion, we resolve the simultaneous estimation of the support regions and the corresponding motion parameters by using fast and simple methods. In the problem of segmenting the

2D rigid moving object, we use the sequential method outlined above. We first estimate the background motion by assuming that it is the dominant motion. Then, after co-registering the images according to this dominant motion, the motion of the moving object is estimated by considering the region formed by the pixels whose registered frame difference is above a threshold. In the problem of inferring 3D rigid structure we used two methods that lead to similar results. The first method simply slides a rectangular window across the image and detects abrupt changes in the motion parameters. The second method uses a quad-tree decomposition. We start by estimating the motion parameters considering the entire image as the support region. The region is recursively decomposed into smaller regions and the motion of each sub-region is estimated. Then we associate regions with similar motion.

In this chapter we study the problem of estimating the motion of the brightness pattern within a given support region. The chapter is organized as follows. In section 3.2 we describe the motion estimation algorithm. We show that the usual approach of minimizing the sum of the square intensity differences between the two co-registered images corresponds to computing the *Maximum Likelihood* (ML) estimate of the unknown parameters when only two frames are taken into account. To estimate the motion parameters we use the method introduced in reference [13]. We discuss the influence of the spatial variability of the brightness pattern on the behavior of the motion estimation algorithm. In section 3.3 we study the estimation error. We derive an expression for the variance of the estimation error and discuss how to use this result to measure the accuracy of the estimates of the motion parameters. Sections 3.4 and 3.5 particularize the study of sections 3.2 and 3.3 to two special motion models: the translational model and the affine model. These two motion models have been widely used in practice. Our work uses the translational motion model to estimate trajectories of feature points, when inferring 3D structure, as described in Part II of the thesis. The affine motion model was used to compute the 2D parallel motions involved in the segmentation of the 2D rigid object and also to estimate the motion of planar surfaces in inferring 3D structure. Section 3.6

summarizes the chapter.

3.2 Motion Estimation

Consider the pair of images $\{\mathbf{I}_1, \mathbf{I}_2\}$. Our goal is the estimation of the motion of the brightness pattern between images \mathbf{I}_1 and \mathbf{I}_2 in a given region \mathcal{R} of the image plane. We parameterize the two-dimensional (2D) motion. We estimate the 2D motion parameters by computing the *Maximum Likelihood* (ML) estimate of all parameters involved.

The ML cost function is a special case of the one introduced in expression (2.3), see section 2.4. Considering only the frames \mathbf{I}_1 and \mathbf{I}_2 , expression (2.3) is written as

$$C_2 = \iint_{\mathcal{R}} \left\{ [\mathbf{I}_1(x, y) - \mathbf{B}(x, y)]^2 + [\mathbf{I}_2(x, y) - \mathcal{M}(\mathbf{p}^\#)\mathbf{B}(x, y)]^2 \right\} dx dy, \quad (3.1)$$

where the integral is over the support region \mathcal{R} . Without loss of generality, we assume in expression (3.1) that the region \mathcal{R} belongs to the background (the same reasoning can be made for a region that belongs to the moving object). The image \mathbf{B} is the unknown real world brightness pattern. The unknown 2D motion is represented by the vector \mathbf{p} that parameterizes the motion operator $\mathcal{M}(\mathbf{p}^\#)$, as introduced in section 2.2. Note that expression (3.1) takes the image \mathbf{I}_1 as the reference, i.e., we have $\mathcal{M}(\mathbf{p}_1^\#)\mathbf{B} = \mathbf{B}$ and $\mathcal{M}(\mathbf{p}_2^\#)\mathbf{B} = \mathcal{M}(\mathbf{p}^\#)\mathbf{B}$ according to the notation introduced in expression (2.3), section 2.4.

To minimize the ML cost function C_2 , given by expression (3.1), with respect to the unknowns \mathbf{B} and \mathbf{p} , we first express the estimate $\hat{\mathbf{B}}$ of \mathbf{B} in terms of the unknown vector \mathbf{p} . Then we insert the estimate $\hat{\mathbf{B}}$ of the real brightness pattern into expression (3.1) and minimize with respect to the motion parameter vector \mathbf{p} .

The ML estimate of the real brightness pattern is the average of the brightness pattern of the two images, after registering \mathbf{I}_2 according to the image motion,

$$\hat{\mathbf{B}} = \frac{1}{2} [\mathbf{I}_1(x, y) + \mathcal{M}(\mathbf{p})\mathbf{I}_2(x, y)]. \quad (3.2)$$

Replacing \mathbf{B} in expression (3.1) by $\hat{\mathbf{B}}$ given by expression (3.2), we obtain, after simple

algebraic manipulations,

$$C_2 = \frac{1}{2} \iint_{\mathcal{R}} \left[\mathbf{I}_1(x, y) - \mathcal{M}(\mathbf{p})\mathbf{I}_2(x, y) \right]^2 dx dy. \quad (3.3)$$

Expression (3.3) shows that the ML estimate of the motion parameter vector is the one that best aligns \mathbf{I}_2 with \mathbf{I}_1 in the *Least Squares* (LS) sense.

To make explicit the warping of image \mathbf{I}_2 according to the motion operator \mathbf{p} , we rewrite $\mathcal{M}(\mathbf{p})\mathbf{I}_2(x, y)$ as

$$\mathcal{M}(\mathbf{p})\mathbf{I}_2(x, y) = \mathbf{I}_2\left(f_x(\mathbf{p}; x, y), f_y(\mathbf{p}; x, y)\right) = \mathbf{I}_2\left(x + d_x(\mathbf{p}; x, y), y + d_y(\mathbf{p}; x, y)\right), \quad (3.4)$$

and the estimate $\hat{\mathbf{p}}$ of the motion vector \mathbf{p} as

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p}} E(\mathbf{p}), \quad \text{where} \quad E(\mathbf{p}) = \iint_{\mathcal{R}} e^2(\mathbf{p}; x, y) dx dy, \quad (3.5)$$

$$\text{and} \quad e(\mathbf{p}; x, y) = \mathbf{I}_1(x, y) - \mathbf{I}_2\left(x + d_x(\mathbf{p}; x, y), y + d_y(\mathbf{p}; x, y)\right). \quad (3.6)$$

In expressions (3.4) and (3.5), the displacement of the pixel (x, y) between images \mathbf{I}_1 and \mathbf{I}_2 is denoted by $\mathbf{d}(\mathbf{p}; x, y) = [d_x(\mathbf{p}; x, y), d_y(\mathbf{p}; x, y)]^T$.

To minimize E in expression (3.5) we use a known technique introduced in reference [13]. This technique uses a *Gauss-Newton method*. The estimate $\hat{\mathbf{p}}$ is computed by refining a previous estimate \mathbf{p}_0 . When dealing with consecutive frames, the initial estimate corresponds to the identity mapping, i.e., $d_x(\mathbf{p}_0; x, y) = 0$ and $d_y(\mathbf{p}_0; x, y) = 0$; when computing the motion between non-consecutive frames, for example frames \mathbf{I}_1 and \mathbf{I}_f , the initial estimate is the motion between frames \mathbf{I}_1 and \mathbf{I}_{f-1} previously computed, i.e., $\mathbf{p}_0 = \hat{\mathbf{p}}_{f-1}$. Since the Gauss-Newton method approximates the error function $e(\mathbf{p})$ by the truncated Taylor series expansion of $e(\mathbf{p}_0 + \boldsymbol{\delta}_p)$, the initial estimate \mathbf{p}_0 must lie in a tight neighborhood of the actual value of the vector \mathbf{p} . This means that the motion between consecutive frames must be small, typically sub-pixel motion. To cope with larger displacements, a spatial pyramid is used. The motion is first computed for the coarsest level of resolution, and then it is propagated as initial estimate to the immediately finer

level. We outline the Gauss-Newton minimization procedure to make self-contained the analysis of the 2D motion estimation algorithm presented in this chapter.

The error function $e(\mathbf{p}; x, y)$ is approximated by neglecting second and higher order terms of the Taylor series expansion of $e(\mathbf{p}_0 + \boldsymbol{\delta}_p)$,

$$e(\mathbf{p}; x, y) \simeq e(\mathbf{p}_0; x, y) + \boldsymbol{\delta}_p^T \nabla_{\mathbf{p}} e(\mathbf{p}_0; x, y), \quad (3.7)$$

where $\boldsymbol{\delta}_p = \mathbf{p} - \mathbf{p}_0$ and $\nabla_{\mathbf{p}} e(\mathbf{p}_0; x, y)$ is the gradient of $e(\mathbf{p}; x, y)$ with respect to \mathbf{p} evaluated at $\mathbf{p} = \mathbf{p}_0$.

After inserting the first-order approximation of expression (3.7) into the cost function (3.5), the estimate $\hat{\mathbf{p}}$ is given by

$$\hat{\mathbf{p}} = \mathbf{p}_0 + \hat{\boldsymbol{\delta}}_p, \quad \text{with} \quad \hat{\boldsymbol{\delta}}_p = \arg \min_{\boldsymbol{\delta}_p} E(\mathbf{p}_0 + \boldsymbol{\delta}_p). \quad (3.8)$$

Equating to $\mathbf{0}$ the gradient of $E(\mathbf{p}_0 + \boldsymbol{\delta}_p)$ with respect to $\boldsymbol{\delta}_p$, we get the estimate $\hat{\boldsymbol{\delta}}_p$ as the solution of the linear system

$$\boldsymbol{\Gamma}_{\mathcal{R}}(\mathbf{p}_0) \hat{\boldsymbol{\delta}}_p = \boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0), \quad (3.9)$$

$$\text{where} \quad \boldsymbol{\Gamma}_{\mathcal{R}}(\mathbf{p}_0) = \iint_{\mathcal{R}} \nabla_{\mathbf{p}} e(\mathbf{p}_0; x, y) \nabla_{\mathbf{p}} e^T(\mathbf{p}_0; x, y) dx dy, \quad (3.10)$$

$$\text{and} \quad \boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0) = - \iint_{\mathcal{R}} e(\mathbf{p}_0; x, y) \nabla_{\mathbf{p}} e(\mathbf{p}_0; x, y) dx dy. \quad (3.11)$$

Denoting by N_p the dimension of the unknown vector \mathbf{p} of motion parameters, the vector $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0)$ has the same dimension of \mathbf{p} , i.e., $N_p \times 1$ and the matrix $\boldsymbol{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ is square $N_p \times N_p$.

The error $e(\mathbf{p}_0; x, y)$ and its gradient $\nabla_{\mathbf{p}} e(\mathbf{p}_0; x, y)$ are computed from the spatial and temporal derivatives of the brightness pattern as

$$e(\mathbf{p}_0; x, y) = -i_t(\mathbf{p}_0; x, y), \quad (3.12)$$

$$\begin{aligned} \nabla_{\mathbf{p}} e(\mathbf{p}_0; x, y) &= -i_x(x, y) \nabla_{\mathbf{p}} d_x(\mathbf{p}_0; x, y) - i_y(x, y) \nabla_{\mathbf{p}} d_y(\mathbf{p}_0; x, y) \\ &= -\nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_0; x, y) \mathbf{i}_{xy}(x, y), \end{aligned} \quad (3.13)$$

where $i_t(\mathbf{p}_0; x, y)$ is the temporal derivative computed by

$$i_t(\mathbf{p}_0; x, y) = \mathbf{I}_2 \left(x + d_x(\mathbf{p}_0; x, y), y + d_y(\mathbf{p}_0; x, y) \right) - \mathbf{I}_1(x, y), \quad (3.14)$$

and $i_x(x, y)$ and $i_y(x, y)$ are the spatial derivatives computed from the reference image $\mathbf{I}_1(x, y)$. The 2×1 vector $\mathbf{i}_{xy}(x, y)$ is defined as

$$\mathbf{i}_{xy}(x, y) = \begin{bmatrix} i_x(x, y) \\ i_y(x, y) \end{bmatrix}, \quad (3.15)$$

and the $N_p \times 2$ matrix $\nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_0; x, y)$ is defined as

$$\nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_0; x, y) = \begin{bmatrix} \nabla_{\mathbf{p}} d_x(\mathbf{p}_0; x, y) & \nabla_{\mathbf{p}} d_y(\mathbf{p}_0; x, y) \end{bmatrix}. \quad (3.16)$$

By replacing expressions (3.12) and (3.13) into the definitions (3.10) and (3.11), we express $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ and $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0)$ in terms of the image derivatives and displacement derivatives. We obtain

$$\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0) = \iint_{\mathcal{R}} \nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_0) \mathbf{i}_{xy} \mathbf{i}_{xy}^T \nabla_{\mathbf{p}} \mathbf{d}(\mathbf{p}_0) dx dy, \quad (3.17)$$

$$\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0) = - \iint_{\mathcal{R}} i_t(\mathbf{p}_0) \nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_0) \mathbf{i}_{xy} dx dy, \quad (3.18)$$

where we omitted the dependence of the integrands on (x, y) for simplicity.

In order to obtain a reliable convergence of the Gauss-Newton method, the equation system (3.9) must be well conditioned, i.e., the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$, given by expression (3.17), must be well conditioned with respect to inversion. A widely used measure for the sensitivity of the solution of the linear system is the *condition number* of the square matrix involved, see reference [27]. The relative error of the solution of a linear system $\mathbf{Ax} = \mathbf{b}$ is approximated by the condition number $k(\mathbf{A})$ of the square matrix \mathbf{A} times the relative errors in \mathbf{A} and \mathbf{b} . The condition number depends on the underlying norm used to measure the error. With the common choice of the matrix 2-norm, the condition number of a matrix is given by the quotient of the largest singular value by the smallest singular value, see reference [27]. Since the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ is symmetric and semi-positive definite, their eigenvalues are positive real and coincide with the singular values. The sensitivity

of the iterates of the motion estimation algorithm are measured by

$$k(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)) = \frac{\lambda_1(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0))}{\lambda_N(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0))}, \quad (3.19)$$

where $\lambda_1(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0))$ is the largest eigenvalue of $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ and $\lambda_N(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0))$ is its smallest eigenvalue.

If the condition number $k(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0))$ is large, the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ is said to be ill-conditioned. In this case, the Gauss-Newton iterates are very sensitive to the noise and the process can not be guaranteed to converge. We will see in sections 3.4 and 3.5 what are the practical implications of requiring the condition number $k(\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0))$ to be small. We will discuss there the difficulty of estimating the motion parameters in terms of the variability of image brightness pattern within the support region \mathcal{R} .

3.3 Estimation Error

This section studies the statistics of the error in estimating the vector \mathbf{p} of motion parameters. This analysis is local, in the sense that we assume small deviations between the true value of the vector of motion parameters and its estimate. This local analysis is very common in estimation problems. It leads, for example, to the establishment of fundamental bounds like the *Cramér-Rao lower bound* (CRB) for the variance of the estimation error, see reference [62].

In our case, due to the specific structure of the estimator, the small deviation assumption enables the derivation of an expression for the expected noise variance in terms of image spatial gradients. The statistics that we obtain are valid in practice as good approximations to the real statistics if the estimation problem is well conditioned, i.e., if the observations, regardless of the noise level, contain “enough information” to estimate the desired parameters (this imprecise definition can be made precise in terms of the usual signal to noise ratio parameter). This situation is the one in which we are interested in because we only use the motion estimates when the corresponding estimation problem is well conditioned in the sense discussed in the previous section.

We denote the actual value of the vector of motion parameters by \mathbf{p}_a . The estimate $\hat{\mathbf{p}}$ is written in terms of a small deviation relative to the actual \mathbf{p}_a . By proceeding in a similar way as done in the previous section, using \mathbf{p}_a instead of \mathbf{p}_0 as the central point of the Taylor series expansion, we obtain

$$\hat{\mathbf{p}} = \mathbf{p}_a + \mathbf{\Gamma}_{\mathcal{R}}^{-1}(\mathbf{p}_a) \boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a), \quad (3.20)$$

where, we recall, the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_a)$ and the vector $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a)$ are given by expressions (3.17) and (3.18) with \mathbf{p}_a instead of \mathbf{p}_0 . The random variable $\hat{\mathbf{p}}$ in expression (3.20) is a non-linear function of the image derivatives $\{i_t(\mathbf{p}_a), \mathbf{i}_{xy} = [i_x, i_y]^T\}$.

The derivatives i_t and \mathbf{i}_{xy} are random variables – they are noisy versions of the actual values of the scene brightness derivatives. The actual value of $i_t(\mathbf{p}_a)$ is $i_{ta}(\mathbf{p}_a) = 0$ because \mathbf{p}_a is the actual value of the vector of motion parameters. The actual value of \mathbf{i}_{xy} is denoted by $\mathbf{i}_{xya} = [i_{xa}, i_{ya}]^T$. Since the image noise $\mathbf{W}_f(x, y)$ is zero mean, the noise corrupting the derivatives i_t , i_x , and i_y is also zero mean. Furthermore, the noise corrupting the temporal derivative i_t is white because the noise images $\mathbf{W}_1(x, y)$ and $\mathbf{W}_2(x, y)$ are independent. The variance of the noise corrupting the image derivatives is denoted by σ_t^2 for i_t , σ_x^2 for i_x , and σ_y^2 for i_y .

We find the expected value of the estimate $\hat{\mathbf{p}}$ by computing the mean of expression (3.20) with respect to the noise of the image derivatives. For small deviations, the first-order approximation of $E\{\hat{\mathbf{p}}\}$ is given by the value of expression (3.20) evaluated at the mean values of the random variables $i_t(\mathbf{p}_a)$ and \mathbf{i}_{xy} , see appendix A, section A.1. Since the mean of $i_t(\mathbf{p}_a)$ is zero, we get $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a) = \mathbf{0}$ and the mean of the estimate $\hat{\mathbf{p}}$ is

$$E\{\hat{\mathbf{p}}\} = \mathbf{p}_a + E\{\hat{\boldsymbol{\delta}}_p\} = \mathbf{p}_a. \quad (3.21)$$

Expression (3.21) states that, to first-order approximation, the estimate $\hat{\mathbf{p}}$ is unbiased.

The covariance matrix of the estimating error, denoted by $\boldsymbol{\Sigma}_p$, is defined as

$$\boldsymbol{\Sigma}_p = E\left\{(\hat{\mathbf{p}} - \mathbf{p}_a)(\hat{\mathbf{p}} - \mathbf{p}_a)^T\right\}. \quad (3.22)$$

The first-order approximation of the covariance matrix Σ_p is related to the partial derivatives of the estimate $\hat{\mathbf{p}}$ with respect to the random variables involved, i.e., with respect to i_t , i_x , and i_y , and to the variances of those random variables. That result, known from estimation theory, see appendix A, section A.2, states that the first-order approximation of the covariance matrix of a random vector $\hat{\mathbf{p}}$ that depends on a set of random variables $\{v_i, i \in V\}$ is given by

$$\Sigma_p = \sum_{k,l \in V} \mathbb{E} \{ (v_k - \overline{v_k}) (v_l - \overline{v_l}) \} \frac{\partial \hat{\mathbf{p}}}{\partial v_k} \frac{\partial \hat{\mathbf{p}}^T}{\partial v_l}, \quad (3.23)$$

where $\overline{v_i}$ denotes the mean of v_i and the partial derivatives are evaluated at $\{v_i = \overline{v_i}, i \in V\}$.

From expressions (3.20), (3.17), and (3.18), we compute the partial derivatives of $\hat{\mathbf{p}}$ with respect to the random variables $i_t(x, y)$, $i_x(x, y)$, and $i_y(x, y)$, and evaluate them at the mean values $i_t(x, y) = i_{ta}(x, y) = 0$, $i_x(x, y) = i_{xa}(x, y)$, $i_y(x, y) = i_{ya}(x, y)$. We get

$$\frac{\partial \hat{\mathbf{p}}}{\partial i_t(x, y)} = -\frac{\partial [\Gamma_{\mathcal{R}}^{-1}(\mathbf{p}_a) \boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a)]}{\partial i_t(x, y)} = \Gamma_{\mathcal{R}}^{-1}(\mathbf{p}_a) \nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_a) \mathbf{i}_{xya}(x, y), \quad (3.24)$$

$$\frac{\partial \hat{\mathbf{p}}}{\partial i_x(x, y)} = -\frac{\partial [\Gamma_{\mathcal{R}}^{-1}(\mathbf{p}_a) \boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a)]}{\partial i_x(x, y)} = \mathbf{0}, \quad (3.25)$$

$$\frac{\partial \hat{\mathbf{p}}}{\partial i_y(x, y)} = -\frac{\partial [\Gamma_{\mathcal{R}}^{-1}(\mathbf{p}_a) \boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a)]}{\partial i_y(x, y)} = \mathbf{0}, \quad (3.26)$$

where the last two are zero because $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_a) = \mathbf{0}$, since the mean value of i_t is zero.

Since we use a continuous representation of the spatial variables x and y , we write the continuous version of expression (3.23). Using the fact that the noise corrupting i_t is white and noting that the derivatives in expressions (3.25) and (3.26) are zero, we obtain for the covariance Σ_p ,

$$\Sigma_p = \sigma_t^2 \iint_{\mathcal{R}} \frac{\partial \hat{\mathbf{p}}}{\partial i_t} \frac{\partial \hat{\mathbf{p}}^T}{\partial i_t} dx dy. \quad (3.27)$$

After replacing the derivative of $\hat{\mathbf{p}}$ with respect to $i_t(x, y)$ given by (3.24), we get

$$\Sigma_p = \sigma_t^2 \Gamma_{\mathcal{R}}^{-1}(\mathbf{p}_a) \iint_{\mathcal{R}} \nabla_{\mathbf{p}} \mathbf{d}^T(\mathbf{p}_a) \mathbf{i}_{xya} \mathbf{i}_{xya}^T \nabla_{\mathbf{p}} \mathbf{d}(\mathbf{p}_a) dx dy \Gamma_{\mathcal{R}}^{-T}(\mathbf{p}_a). \quad (3.28)$$

Noting that the integral above is the matrix $\Gamma_{\mathcal{R}}$ evaluated at \mathbf{p}_a (compare to expression (3.17)), and that the matrix $\Gamma_{\mathcal{R}}(\mathbf{p}_a)$ is symmetric, we obtain for the error covariance

$$\Sigma_p = \sigma_t^2 \Gamma_{\mathcal{R}}^{-1}(\mathbf{p}_a). \quad (3.29)$$

Expression (3.29) provides an inexpensive way to compute the reliability of the motion estimates. The matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_a)$ is in general unknown because it depends on the actual value \mathbf{p}_a of the unknown vector \mathbf{p} . Obviously, the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_a)$ can be approximated by the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ used in the iterative estimation algorithm. We note that when the motion model is linear in the motion parameters, as it is the case with the majority of motion models used in practice, the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p})$ becomes independent of the vector \mathbf{p} because the derivatives of the displacement $\mathbf{d}(\mathbf{p})$ involved in expression (3.17) do not depend on the motion parameters. In this case, the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ does not change along the iterative estimation algorithm. The matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ depends uniquely on the image region \mathcal{R} and $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ will be denoted simply by $\mathbf{\Gamma}_{\mathcal{R}}$. Since the noise variance σ_t^2 is considered to be constant, we measure the error covariance for different regions by comparing the corresponding matrices $\mathbf{\Gamma}_{\mathcal{R}}^{-1}$. For example, the mean square Euclidean distance between the true vector \mathbf{p}_a and the estimated vector $\hat{\mathbf{p}}$, denoted by σ_p^2 , is proportional to the trace of the matrix $\mathbf{\Gamma}_{\mathcal{R}}^{-1}$,

$$\sigma_p^2 = \mathbb{E} \left\{ (\hat{\mathbf{p}} - \mathbf{p}_a)^T (\hat{\mathbf{p}} - \mathbf{p}_a) \right\} = \sigma_t^2 \text{tr} (\mathbf{\Gamma}_{\mathcal{R}}^{-1}). \quad (3.30)$$

These concepts will become clearer in the next two sections where we particularize for the translational motion model and to the affine motion model the results derived in this section and in the previous section.

3.4 Translational Motion

This section studies the translational motion model. The translational motion model is characterized by a constant displacement for all the pixels that fall into the region \mathcal{R} . We use the translational motion model to estimate the displacement of pointwise features when inferring three-dimensional (3D) structure from two-dimensional (2D) motion, see Part II, chapter 8. In this case the region \mathcal{R} is a small square centered about the coordinates of the feature. The translational model is also frequently used to represent the motion within a larger region \mathcal{R} for special cases of the 3D shape that is projected

into \mathcal{R} and for special cases of the 3D motion of the camera. This happens in a special case of the *Generative Video* (GV) [34, 35, 36, 37] framework. GV assumes 2D parallel motions, as described in chapter 2; if the motions are further restricted to be translations, the 2D translational motion model studied in this section can be used to estimate the position vectors.

For the translational motion model, the vector \mathbf{p} of motion parameters is defined as

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}, \quad (3.31)$$

where p_1 and p_2 determine the displacement $\mathbf{d}(\mathbf{p})$ as

$$\mathbf{d}(\mathbf{p}) = \begin{bmatrix} d_x(\mathbf{p}) \\ d_y(\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \mathbf{p}. \quad (3.32)$$

Motion estimation

The motion parameters are estimated by particularizing to the model of expression (3.32) the algorithm described in section 3.2.

To compute the matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ and the vector $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0)$ needed for the Gauss-Newton iterates, we start by making explicit the gradient of the displacement \mathbf{d} with respect to the vector \mathbf{p} ,

$$\nabla_{\mathbf{p}} \mathbf{d}^T = \begin{bmatrix} \nabla_{\mathbf{p}} d_x & \nabla_{\mathbf{p}} d_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}_{2 \times 2}. \quad (3.33)$$

As advanced at the end of the previous section, the gradient in expression (3.33) does not depend on the vector \mathbf{p} . For this reason, the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ will be independent of \mathbf{p}_0 and will remain constant along the iterative process. By replacing expression (3.33) into expressions (3.17) and (3.18), we get

$$\mathbf{\Gamma}_{\mathcal{R}} = \iint_{\mathcal{R}} \mathbf{i}_{xy} \mathbf{i}_{xy}^T dx dy = \begin{bmatrix} \iint_{\mathcal{R}} i_x^2 dx dy & \iint_{\mathcal{R}} i_x i_y dx dy \\ \iint_{\mathcal{R}} i_x i_y dx dy & \iint_{\mathcal{R}} i_y^2 dx dy \end{bmatrix}, \quad (3.34)$$

$$\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0) = - \iint_{\mathcal{R}} i_t(\mathbf{p}_0) \mathbf{i}_{xy} dx dy = - \begin{bmatrix} \iint_{\mathcal{R}} i_x i_t(\mathbf{p}_0) dx dy \\ \iint_{\mathcal{R}} i_y i_t(\mathbf{p}_0) dx dy \end{bmatrix}. \quad (3.35)$$

Each iteration of the algorithm updates the initial estimate \mathbf{p}_0 as $\hat{\mathbf{p}} = \mathbf{p}_0 + \hat{\boldsymbol{\delta}}_p$ with $\hat{\boldsymbol{\delta}}_p$ obtained from expression (3.9) with $\mathbf{\Gamma}_{\mathcal{R}}$ and $\boldsymbol{\gamma}_{\mathcal{R}}$ given by expressions (3.34) and (3.35).

The behavior of the estimation algorithm depends on the condition number of the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ of expression (3.34). The condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ was defined in expression (3.19) for a general motion model. For the translational motion model, it is the quotient of the first eigenvalue of $\mathbf{\Gamma}_{\mathcal{R}}$ by the second eigenvalue,

$$k(\mathbf{\Gamma}_{\mathcal{R}}) = \frac{\lambda_1(\mathbf{\Gamma}_{\mathcal{R}})}{\lambda_2(\mathbf{\Gamma}_{\mathcal{R}})}, \quad (3.36)$$

where $\lambda_1(\mathbf{\Gamma}_{\mathcal{R}})$ and $\lambda_2(\mathbf{\Gamma}_{\mathcal{R}})$ are the eigenvalues of $\mathbf{\Gamma}_{\mathcal{R}}$ in expression (3.34) with $\lambda_1(\mathbf{\Gamma}_{\mathcal{R}}) \geq \lambda_2(\mathbf{\Gamma}_{\mathcal{R}})$. If the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is large, the Gauss-Newton iterates are very sensitive to the noise and the process can not be guaranteed to converge. If the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is small, i.e., if it is close to unity, since $k(\mathbf{\Gamma}_{\mathcal{R}}) \geq 1$, the linear system involved in the Gauss-Newton method is well conditioned.

We discuss when $k(\mathbf{\Gamma}_{\mathcal{R}})$ has large values. To see the influence of the image brightness pattern within region \mathcal{R} on the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ consider that $\iint_{\mathcal{R}} i_x i_y dx dy = 0$. The matrix $\mathbf{\Gamma}_{\mathcal{R}}$ becomes diagonal and the condition number is simply

$$k(\mathbf{\Gamma}_{\mathcal{R}}) = \frac{\iint_{\mathcal{R}} i_x^2 dx dy}{\iint_{\mathcal{R}} i_y^2 dx dy} \quad (3.37)$$

$$\text{if} \quad \iint_{\mathcal{R}} i_x^2 dx dy \geq \iint_{\mathcal{R}} i_y^2 dx dy \quad (3.38)$$

or the inverse if the inequality goes in the opposite way. If one of the components of the spatial image gradient is much larger than the other, $k(\mathbf{\Gamma}_{\mathcal{R}})$ becomes large and the equation system (3.9) is ill-conditioned. The condition

$$k(\mathbf{\Gamma}_{\mathcal{R}}) < \lambda, \quad (3.39)$$

where λ is a threshold, restricts the brightness pattern within region \mathcal{R} not to have variability along some direction much higher than the variability along the perpendicular direction.

The analysis in the paragraph above explains the well known *aperture problem*. The aperture problem is usually described as the impossibility of estimating locally the 2D motion. In fact, if the region \mathcal{R} contains a single pixel, the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ given by expres-

sion (3.34) is singular; we obtain $\det(\mathbf{\Gamma}_{\mathcal{R}}) = 0$ by removing the integrals from expression (3.34), and the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is $+\infty$. This happens because the two motion parameters can not be determined by the single constraint imposed by the brightness constancy.

The study of the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ shows that, for particular structures of the brightness pattern, it is very difficult to estimate the 2D motion, even when the region \mathcal{R} contains several pixels.

Expression (3.37) was obtained with $\iint_{\mathcal{R}} i_x i_y dx dy = 0$. We should note, however, that the case where

$$\iint_{\mathcal{R}} i_x i_y dx dy \neq 0 \quad (3.40)$$

does not correspond to a more general situation. In fact, it can be shown that an appropriate rotation of the brightness pattern makes

$$\iint_{\mathcal{R}} i_x i_y dx dy = 0, \quad (3.41)$$

without changing the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ – as we would expect, the conditioning of the estimation of the motion of the brightness pattern is independent of 2D rigid transformations of the brightness pattern.

Figure 3.1 illustrates the dependence of the conditioning of the 2D motion estimation on the structure of the brightness pattern. For each of the eight 10×10 images in Figure 3.1, we determine the condition number of the matrix $\mathbf{\Gamma}_{\mathcal{R}}$. The condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$, obtained by evaluating expression (3.36), is on the right side of each image in Figure 3.1. The texture of the brightness pattern shown on the top left image is such that there is no dominant direction over the entire region \mathcal{R} . We expect that the estimation of the 2D motion of a pattern of this kind is very well conditioned. In fact, over the entire image no component of the spatial gradient dominates, and the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ captures this behavior. We get $k(\mathbf{\Gamma}_{\mathcal{R}}) = 1.34$ – the value of $k(\mathbf{\Gamma}_{\mathcal{R}})$ is close to unity indicating that the linear system involved in the Gauss-Newton iterates of the motion estimation algorithm is well conditioned. In contrast to this case, the texture of

the brightness pattern shown in the bottom right image of Figure 3.1 exhibits a clear dominant direction. It is very hard to perceive the 2D motion of these type of patterns because only the component of the motion that is perpendicular to the dominant direction of the texture is perceived. The condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ captures the indetermination in estimating the 2D motion – it is $k(\mathbf{\Gamma}_{\mathcal{R}}) = 133.82$ indicating that the linear system involved in the Gauss-Newton iterates of the motion estimation algorithm is ill-conditioned. The other images of Figure 3.1 illustrate intermediate cases.

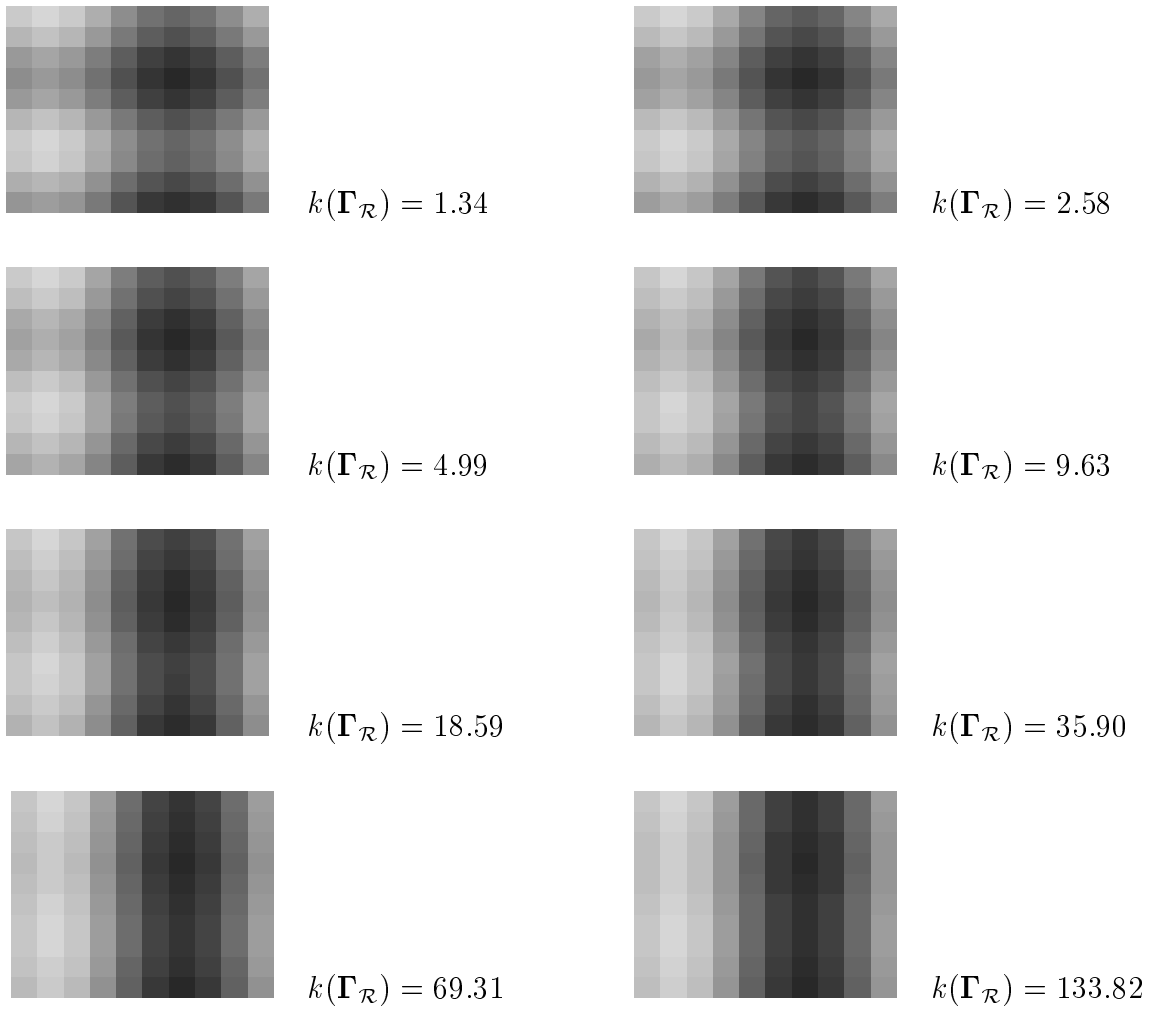


Figure 3.1: The dependence of the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ of the matrix involved in the motion estimation algorithm on the structure of the image brightness pattern. When the texture of the brightness pattern exhibits a dominant direction, the motion estimation is ill conditioned – see the bottom right image and the high value of $k(\mathbf{\Gamma}_{\mathcal{R}})$.

Estimation error

The covariance matrix of the estimation error for the translational motion model is given by expression (3.29) after replacing $\mathbf{\Gamma}_{\mathcal{R}}$ by expression (3.34),

$$\mathbf{\Sigma}_p = \sigma_t^2 \mathbf{\Gamma}_{\mathcal{R}}^{-1} = \sigma_t^2 \begin{bmatrix} \iint_{\mathcal{R}} i_x^2 dx dy & \iint_{\mathcal{R}} i_x i_y dx dy \\ \iint_{\mathcal{R}} i_x i_y dx dy & \iint_{\mathcal{R}} i_y^2 dx dy \end{bmatrix}^{-1}. \quad (3.42)$$

The knowledge of the error covariance matrix $\mathbf{\Sigma}_p$ enables us to compute the reliability of a displacement estimate in an easy way. In fact, in section 3.3, we saw that the mean square error of the displacement estimate (in the sense of the Euclidean distance) is the trace of the covariance matrix $\mathbf{\Sigma}_p$, see expression (3.30). In terms of image gradients, we get the following expression for the mean square error, denoted by σ_p^2 ,

$$\sigma_p^2 = \sigma_t^2 \frac{\iint_{\mathcal{R}} i_y^2 dx dy + \iint_{\mathcal{R}} i_x^2 dx dy}{\iint_{\mathcal{R}} i_x^2 dx dy \iint_{\mathcal{R}} i_y^2 dx dy - \left(\iint_{\mathcal{R}} i_x i_y dx dy \right)^2}. \quad (3.43)$$

In Part II of the thesis, when recovering 3D structure from 2D motion estimates, we use the estimate of the mean square error σ_p^2 given by expression (3.43) to weight motion estimates corresponding to different regions.

To interpret the mean square error σ_p^2 given by expression (3.43), let us consider again that the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ is diagonal. This is the general case because, as for the condition number, it can be shown that any non-diagonal matrix $\mathbf{\Gamma}_{\mathcal{R}}$ can be made diagonal without changing σ_p^2 , by an appropriate rotation of the image brightness pattern. When $\iint_{\mathcal{R}} i_x i_y dx dy = 0$, the mean square error σ_p^2 is

$$\sigma_p^2 = \sigma_t^2 \left(\frac{1}{\iint_{\mathcal{R}} i_x^2 dx dy} + \frac{1}{\iint_{\mathcal{R}} i_y^2 dx dy} \right). \quad (3.44)$$

Expression (3.44) states that the error in the estimate of the displacement is proportional to the inverse of the sum of the square components of the image gradient within region \mathcal{R} . This coincides with the intuitive notion that the higher the spatial variability of the brightness pattern is, the lower the error in estimating the motion is. As expected, it is also clear that the estimation error decreases when the size of the region \mathcal{R} increases.

Figure 3.2 illustrates the dependence of the expected square error of the 2D motion estimates on the image brightness pattern. To isolate the estimation error from the eventual ill-posedness of the motion estimation problem, we used brightness patterns that do not have a dominant texture direction, i.e., we used brightness patterns for which the linear system involved in the motion estimation is well conditioned. In particular, we used the brightness pattern of the top left image of Figure 3.1 to generate all the images of Figure 3.2 by changing the brightness contrast. The conditioning of the linear system involved in the motion estimation problem does not depend on the brightness contrast, as shown by the constant value of the condition number, $k(\mathbf{\Gamma}_{\mathcal{R}}) = 1.34$ for all the images in Figure 3.2.

For each image in Figure 3.2, we computed the mean square error σ_p^2 by evaluating expression (3.43). Since the goal is to illustrate the influence of the brightness pattern on σ_p^2 , we made $\sigma_t^2 = 1$ when evaluating expression (3.43). The values obtained for σ_p^2 are shown in Figure 3.2 on the right side of the corresponding image. The top left image of Figure 3.2 has a very high brightness contrast. For this reason, we expect that the estimate of the 2D motion of such a pattern is very accurate. In fact, the sum of the square components of the image gradient has a high value and the value of the motion estimation mean square error is low, $\sigma_p^2 = 0.19$. When the brightness contrast decreases, we expect less accurate motion estimates. The values of σ_p^2 in Figure 3.2 are in agreement with this. The expected square error σ_p^2 increases with the decrease of brightness contrast because the square components of the image gradient decrease. The bottom right image of Figure 3.2 shows the extreme situation of a pattern with almost zero brightness contrast. For this pattern, the expected mean square estimation error is very high – larger than 60 times the error for the top left image. It is therefore hopeless to try to compute accurate motion estimates for this kind of low contrast patterns. Note that this is due to the fundamental bound on the motion estimation error, not to the conditioning of the linear system involved in the motion estimation algorithm (the condition number $k(\mathbf{\Gamma}_{\mathcal{R}}) = 1.34$ is close to unity indicating that the linear system is well conditioned).

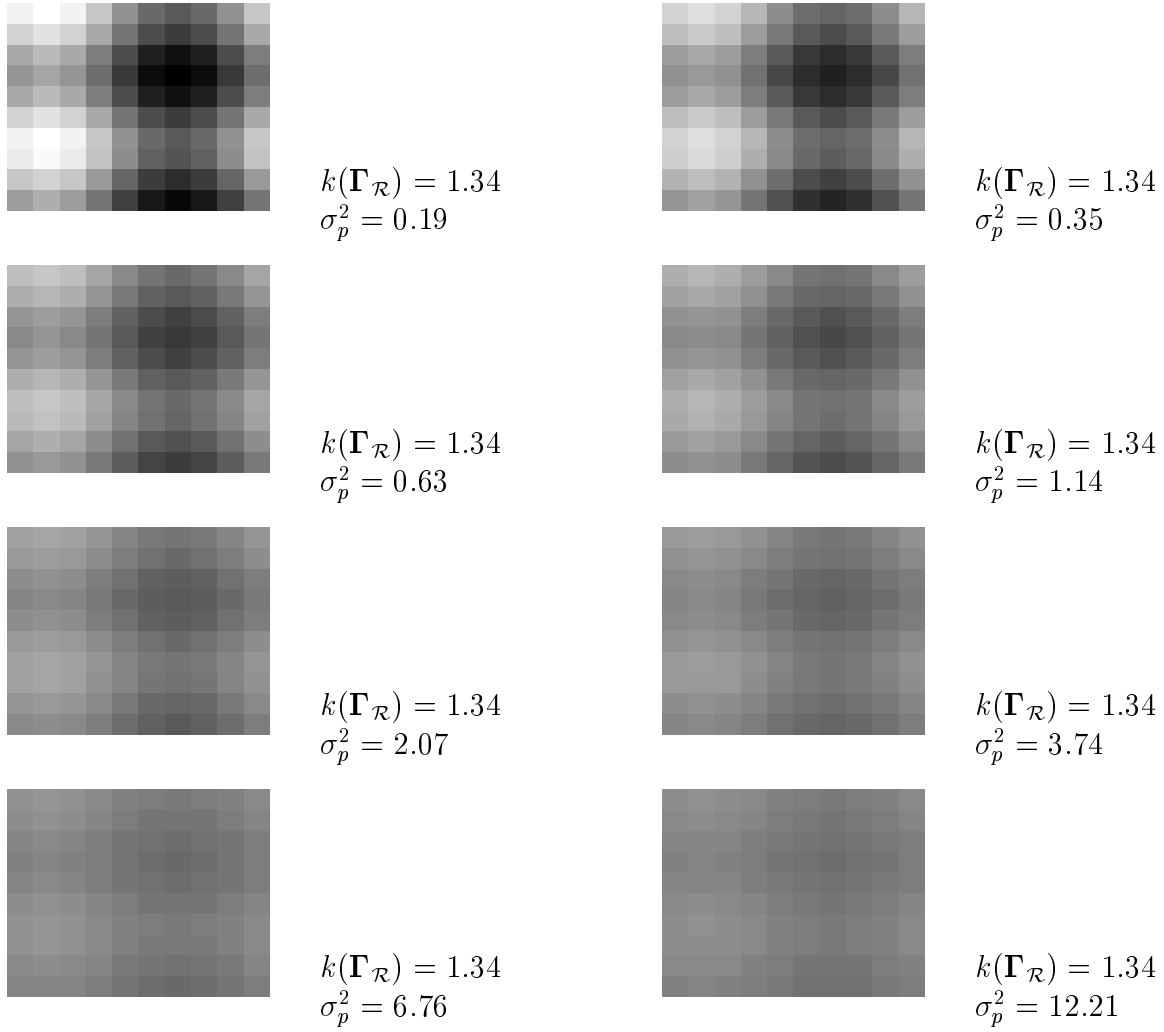


Figure 3.2: The dependence of the motion estimation error on the image brightness pattern. The expected square error σ_p^2 increases with the decrease of the brightness contrast.

In summary, for the estimation algorithm to be stable, the two components of the image gradient should not have too radically different magnitude values. With respect to the mean square error of the displacement estimate, we argued that when the magnitude of the components of the image gradient is large, the error is smaller.

3.5 Affine Motion

This section studies the affine motion model. The affine model is widely used in practice. We use the affine motion model to compute the position vectors involved in the problem of segmenting 2D rigid moving objects, as introduced in chapter 2. In Part II of the thesis we will see that a planar surface moving far away from the camera, undergoing an arbitrary three-dimensional (3D) motion, induces an affine motion of the brightness pattern between pairs of frames. Thus, when inferring 3D structure from 2D motion, as described in Part II, chapter 9, we will also use the affine motion model studied in this section.

The vector \mathbf{p} parameterizing the affine motion model has 6 components,

$$\mathbf{p} = [p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ p_6]^T. \quad (3.45)$$

The affine displacement $\mathbf{d}(\mathbf{p})$, to which we will also refer to as the affine mapping, is given by

$$\mathbf{d}(\mathbf{p}) = \begin{bmatrix} d_x(\mathbf{p}) \\ d_y(\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} p_3 & p_5 \\ p_4 & p_6 \end{bmatrix} \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix}, \quad (3.46)$$

where x_c and y_c are arbitrary constants that in practice we choose to be the center of the region \mathcal{R} to improve the stability of the estimation algorithm, as will become clear below.

Motion estimation

The affine motion parameters are estimated by using the algorithm described in section 3.1. To specialize the expressions of matrix $\mathbf{\Gamma}_{\mathcal{R}}(\mathbf{p}_0)$ and vector $\boldsymbol{\gamma}_{\mathcal{R}}(\mathbf{p}_0)$ involved in the Gauss-Newton iterates to the affine motion model, we start by computing the gradient of the affine displacement with respect to the motion parameters, obtaining

$$\nabla_{\mathbf{p}} \mathbf{d}^T = [\nabla_{\mathbf{p}} d_x \quad \nabla_{\mathbf{p}} d_y] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ x - x_c & 0 \\ 0 & x - x_c \\ y - y_c & 0 \\ 0 & y - y_c \end{bmatrix}. \quad (3.47)$$

As with the translation motion model, because the affine motion model is linear on the motion parameters, the gradient in expression (3.47) does not depend on the vector \mathbf{p} . The matrix $\mathbf{\Gamma}_{\mathcal{R}}$ will then be independent of \mathbf{p}_0 and remain constant along the iterative process. By replacing expression (3.47) into expressions (3.17) and (3.18), we get the following expressions for $\mathbf{\Gamma}_{\mathcal{R}}$ and $\gamma_{\mathcal{R}}(\mathbf{p}_0)$,

$$\mathbf{\Gamma}_{\mathcal{R}} = \begin{bmatrix} \int_{\mathcal{R}} i_x^2 & \int_{\mathcal{R}} i_x i_y & \int_{\mathcal{R}} \tilde{x} i_x^2 & \int_{\mathcal{R}} \tilde{x} i_x i_y & \int_{\mathcal{R}} \tilde{y} i_x^2 & \int_{\mathcal{R}} \tilde{y} i_x i_y \\ \int_{\mathcal{R}} i_x i_y & \int_{\mathcal{R}} i_y^2 & \int_{\mathcal{R}} \tilde{x} i_x i_y & \int_{\mathcal{R}} \tilde{x} i_y^2 & \int_{\mathcal{R}} \tilde{y} i_x i_y & \int_{\mathcal{R}} \tilde{y} i_y^2 \\ \int_{\mathcal{R}} \tilde{x} i_x^2 & \int_{\mathcal{R}} \tilde{x} i_x i_y & \int_{\mathcal{R}} \tilde{x}^2 i_x^2 & \int_{\mathcal{R}} \tilde{x}^2 i_x i_y & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_x^2 & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_x i_y \\ \int_{\mathcal{R}} \tilde{x} i_x i_y & \int_{\mathcal{R}} \tilde{x} i_y^2 & \int_{\mathcal{R}} \tilde{x}^2 i_x i_y & \int_{\mathcal{R}} \tilde{x}^2 i_y^2 & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_x i_y & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_y^2 \\ \int_{\mathcal{R}} \tilde{y} i_x^2 & \int_{\mathcal{R}} \tilde{y} i_x i_y & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_x^2 & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_x i_y & \int_{\mathcal{R}} \tilde{y}^2 i_x^2 & \int_{\mathcal{R}} \tilde{y}^2 i_x i_y \\ \int_{\mathcal{R}} \tilde{y} i_x i_y & \int_{\mathcal{R}} \tilde{y} i_y^2 & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_x i_y & \int_{\mathcal{R}} \tilde{x} \tilde{y} i_y^2 & \int_{\mathcal{R}} \tilde{y}^2 i_x i_y & \int_{\mathcal{R}} \tilde{y}^2 i_y^2 \end{bmatrix}, \quad (3.48)$$

$$\gamma_{\mathcal{R}}(\mathbf{p}_0) = - \begin{bmatrix} \iint_{\mathcal{R}} i_x i_t(\mathbf{p}_0) dx dy \\ \iint_{\mathcal{R}} i_y i_t(\mathbf{p}_0) dx dy \\ \iint_{\mathcal{R}} \tilde{x} i_x i_t(\mathbf{p}_0) dx dy \\ \iint_{\mathcal{R}} \tilde{x} i_y i_t(\mathbf{p}_0) dx dy \\ \iint_{\mathcal{R}} \tilde{y} i_x i_t(\mathbf{p}_0) dx dy \\ \iint_{\mathcal{R}} \tilde{y} i_y i_t(\mathbf{p}_0) dx dy \end{bmatrix}, \quad (3.49)$$

$$\text{where} \quad \begin{cases} \tilde{x} = x - x_c \\ \tilde{y} = y - y_c \end{cases}. \quad (3.50)$$

The stability of the motion estimation algorithm depends on the condition number of the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ above. If the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is high, the linear system involved in the motion estimation iterative algorithm is ill conditioned. If the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is low (close to one) that system is well conditioned and the algorithm is stable. To understand the influence of the constants x_c and y_c on the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$, we evaluated $k(\mathbf{\Gamma}_{\mathcal{R}})$ for a simpler case – the one-dimensional (1D) affine motion model. The condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ for the 2D affine model is complex to study because, in opposition to the translational motion model, the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ in expression (3.48) can not be diagonalized by rotating the image brightness pattern. The condition number of the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ involved in the estimation of the parameters of the 1D affine model exhibits the property we want to illustrate for the 2D affine model and leads to a much simpler expression.

The 1D affine motion model is parameterized by the vector $\mathbf{p} = [p_1, p_2]^T$. The 1D displacement is

$$d(\mathbf{p}) = p_1 + p_2(x - x_c). \quad (3.51)$$

Proceeding as we did for the 2D case, we obtain the 2×2 matrix $\mathbf{\Gamma}_{\mathcal{R}}$ involved in the 1D affine motion estimation,

$$\mathbf{\Gamma}_{\mathcal{R}} = \begin{bmatrix} \int_{\mathcal{R}} i_x^2 dx & \int_{\mathcal{R}} (x - x_c) i_x^2 dx \\ \int_{\mathcal{R}} (x - x_c) i_x^2 dx & \int_{\mathcal{R}} (x - x_c)^2 i_x^2 dx \end{bmatrix}. \quad (3.52)$$

The 2×2 semi-positive definite matrix $\mathbf{\Gamma}_{\mathcal{R}}$ in expression (3.52) is the 1D version of the matrix in expression (3.48). We obtain the condition number of a generic semi-positive definite 2×2 matrix in terms of the entries of the matrix, by expressing the quotient of the larger by the smaller eigenvalue of the matrix,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \implies k(\mathbf{A}) = \frac{a_{11} + a_{22} + \sqrt{(a_{11} - a_{22})^2 + 4a_{12}^2}}{a_{11} + a_{22} - \sqrt{(a_{11} - a_{22})^2 + 4a_{12}^2}}. \quad (3.53)$$

In Figure 3.3 we plot the condition number of the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ in expression (3.52) in terms of the entries of the matrix. We fixed the entry

$$a_{11} = \int_{\mathcal{R}} i_x^2 dx = 1. \quad (3.54)$$

We used expression (3.53) to evaluate the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ with

$$a_{22} = \int_{\mathcal{R}} (x - x_c)^2 i_x^2 dx \quad (3.55)$$

ranging from 1 to 100, and

$$a_{12} = \int_{\mathcal{R}} (x - x_c) i_x^2 dx \quad (3.56)$$

ranging from -5 to 5 . From Figure 3.3 we can see that the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is small if $\int_{\mathcal{R}} (x - x_c)^2 i_x^2 dx$ is close to one and $\int_{\mathcal{R}} (x - x_c) i_x^2 dx$ is close to zero. In this case, the linear system involved in the motion estimation algorithm is very well conditioned. If either the value of $\int_{\mathcal{R}} (x - x_c)^2 i_x^2 dx$ or the absolute value of $\int_{\mathcal{R}} (x - x_c) i_x^2 dx$ are very high, the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ is high and the linear system is ill conditioned. For this reason, the optimal choice for the constant x_c is the center of the region \mathcal{R} .

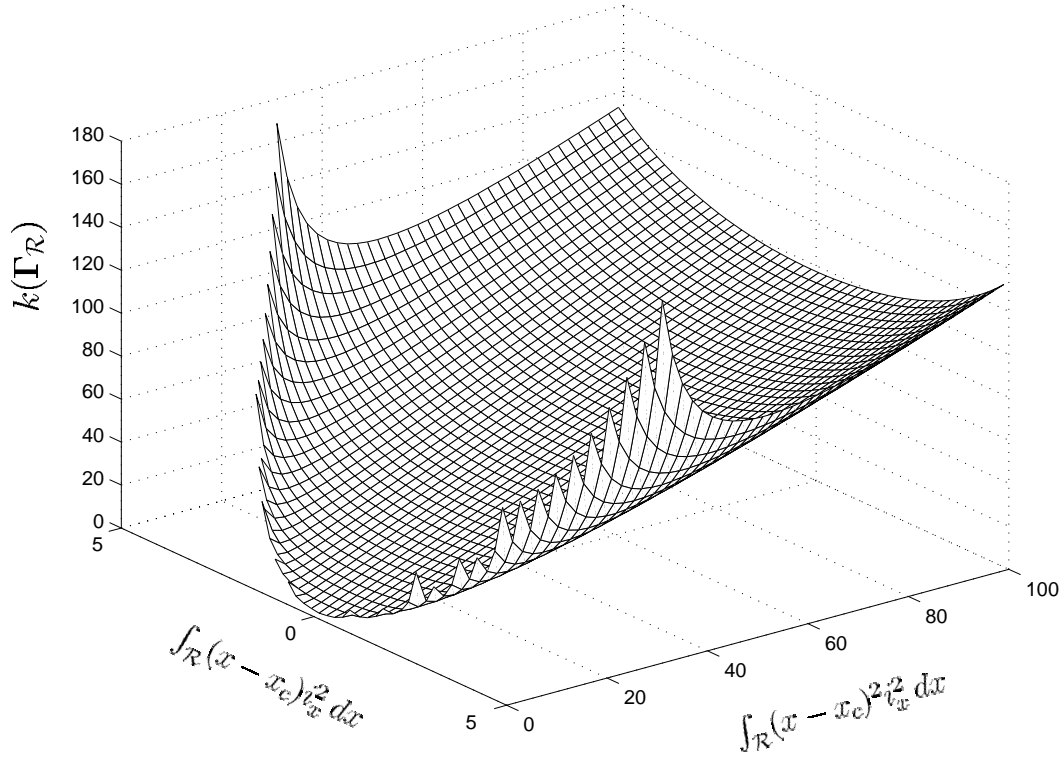


Figure 3.3: Affine motion estimation. The dependence of the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ on the entries (1, 2) and (2, 2) of the matrix $\mathbf{\Gamma}_{\mathcal{R}}$ of expression (3.52). The entry (1, 1) was kept fixed, $\int_{\mathcal{R}} i_x^2 dx = 1$. If the constant x_c is chosen to be far from the center of the region \mathcal{R} , the linear system involved in the motion estimation algorithm may be very ill conditioned.

For the 2D affine motion model, the condition number of the 6×6 matrix $\mathbf{\Gamma}_{\mathcal{R}}$ is given by

$$k(\mathbf{\Gamma}_{\mathcal{R}}) = \frac{\lambda_1(\mathbf{\Gamma}_{\mathcal{R}})}{\lambda_6(\mathbf{\Gamma}_{\mathcal{R}})}, \quad (3.57)$$

where $\lambda_1(\mathbf{\Gamma}_{\mathcal{R}})$ is the largest eigenvalue of $\mathbf{\Gamma}_{\mathcal{R}}$ and $\lambda_6(\mathbf{\Gamma}_{\mathcal{R}})$ is its smallest eigenvalue. We expect the same kind of behavior of the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$ in terms of the constants x_c and y_c . We then choose the constants x_c and y_c to be the center of the support region \mathcal{R} , as noted at the beginning of the section.

Estimation error

The covariance matrix of the estimation error for the affine motion model is given by expression (3.29) after replacing $\mathbf{\Gamma}_{\mathcal{R}}$ by expression (3.48),

$$\mathbf{\Sigma}_p = \sigma_t^2 \mathbf{\Gamma}_{\mathcal{R}}^{-1}. \quad (3.58)$$

The knowledge of the error covariance matrix $\mathbf{\Sigma}_p$ enables us to compute the reliability of a motion parameter estimate in an easy way. In Part II of the thesis, we will use the reliability of the estimates of the motion parameters to weight the contribution of those estimates on the recovery of 3D structure. We define the error $\sigma_{\mathcal{P}}^2$ as the mean square error Euclidean distance between the true and estimated values of a subset of the set of motion parameters,

$$\sigma_{\mathcal{P}}^2 = \mathbb{E} \left\{ \sum_{i \in \mathcal{P}} (\hat{p}_i - p_{ia})^2 \right\}, \quad (3.59)$$

where \mathcal{P} is the subset of the set of motion parameters and p_{ia} is the actual value of parameter p_i . Since the error $\sigma_{\mathcal{P}}^2$ depends only on the main diagonal entries of the covariance matrix $\mathbf{\Sigma}_p$, we obtain from expression (3.58),

$$\sigma_{\mathcal{P}}^2 = \sigma_t^2 \sum_{i \in \mathcal{P}} \mathbf{\Pi}_{ii}, \quad \text{where} \quad \mathbf{\Pi} = \mathbf{\Gamma}_{\mathcal{R}}^{-1}. \quad (3.60)$$

In expression (3.60), the set \mathcal{P} is an arbitrary subset of the parameters of the affine motion model. For example, we can compute the variance of the estimation error of a single parameter p_i by making $\mathcal{P} = \{p_i\}$, or the mean square Euclidean error of the translational component of the affine motion by making $\mathcal{P} = \{p_1, p_2\}$. In Part II, chapter 9, section 9.4, we will be interested in the error $\sigma_{\mathcal{P}}^2$ for particular subsets \mathcal{P} of the set of parameters of the affine motion model. We will show how to use the reliability of the estimates of the motion parameters, measured by $\sigma_{\mathcal{P}}^2$, into the recovery of 3D rigid structure, without additional computational cost.

3.6 Summary

In this chapter we studied the estimation of the motion of the brightness pattern between a pair of frames. This task is crucial for both problems addressed in the thesis – the segmentation of the two-dimensional (2D) rigid moving object, and the inference of three-dimensional (3D) rigid structure.

The motion estimation technique we use was proposed by Bergen, Anandan, Hanna, and Hingorani, in reference [13]. We overviewed their technique in this chapter to make the thesis self-contained. The original contributions of the chapter are the discussion on the conditioning of the 2D motion estimation problem and the computation of the covariance of the estimation error. We discuss in section 3.2 the conditioning of the 2D motion estimation problem in terms of the condition number of the matrix involved in the Gauss-Newton iterates of the method of reference [13]. We derive an expression for the covariance of the estimation error in terms of the image spatial gradients in section 3.3.

In sections 3.4 and 3.5, we specialize the analysis to the two motion models we use in practice – the translational motion model, and the affine motion model. For the translational motion model, we relate the conditioning of the estimation problem to the variability of the spatial brightness pattern. For the affine motion model, we also discuss the influence of the origin of the affine mapping on the conditioning of the estimation problem. For both motion models, we derive expressions for the expected square of the Euclidean distance between the true and estimated values of the parameters.

Chapter 4

Direct Inference of 2D Rigid Shape

4.1 Introduction

In this chapter we develop an algorithm for segmenting a two-dimensional (2D) rigid moving object from an image sequence. Our algorithm is a feasible approximation to the *Maximum Likelihood* (ML) estimation of the unknowns involved in the problem, as introduced in chapter 2. Two features distinguish our method from other approaches. First, we take into account the rigidity of the moving object over a set of frames. Second, we estimate the template of the moving object directly from the image intensity values. We will see that our algorithm recovers accurate templates in challenging contexts such as those when the texture of the object is similar to the texture of the background.

In chapter 2, we introduced the problem of segmenting a 2D rigid object and formulated the ML estimate. In chapter 3 we described how we estimate the 2D motions. In this chapter, we assume that the motions have been correctly estimated and are known. We should note that, in reality, the motions are continuously estimated. Assuming the motions are known, the problem we address in this chapter is the minimization of the ML cost function with respect to the remaining parameters, i.e., with respect to the template of the moving object, the texture of the moving object, and the texture of the background.

We plug-in the 2D motion estimates into the ML cost function. The estimate of the texture of the moving object is easily obtained in terms of the unknown template.

The accurate segmentation of the moving object is achieved by using a two-step iterative algorithm to minimize the ML cost function, after replacing the estimate of the texture of the moving object. The steps of the iterative algorithm are: (i) estimate the background for known object template; (ii) estimate the object template for known background. The algorithm converges in a few iterations. Typically, the estimate of the template of the moving object stabilizes after three to five iterations. The algorithm is computationally very simple, because we find closed-form solutions for the two steps involved. In step (i), the estimate of the background is an average of the appropriate regions of the observed frames. In step (ii), the estimate of the template of the moving object leads to a binary test evaluated at each pixel.

Once the two-step iterative algorithm converges, i.e, when the template estimate stabilizes, if new frames are available, we proceed only with the updating of the textures of the object and of the background. The template estimate is frozen and, as new images become available, the object and background textures are updated by computing recursively their ML estimates.

We illustrate with an experiment the convergence of the two-step iterative method. We apply the segmentation algorithm to one computer generated image sequence using real images. In this experiment, the task of reconstructing the moving object template is particularly challenging due to the complexity of the template.

This chapter is organized as follows. Section 4.2 describes how we initialize the segmentation algorithm. The initialization phase provides an initial estimate of the object template to be used in the next step of the algorithm. The procedure to minimize the ML cost function is described in section 4.3. We detail the two-step iterative algorithm to recover the template of the moving object. In section 4.4 we present an illustrative experiment. Section 4.5 summarizes the chapter.

In chapter 5, we study the behavior of the binary test involved in the segmentation algorithm. Experiments with real video sequences are reported in chapter 6.

4.2 Initialization: Object Mask

To initialize the segmentation algorithm, we need an initial estimate of the background. A simple, often used, estimate for the background is the average of the images in the sequence, including or not a robust statistic technique like outlier rejection, see for example reference [41]. The quality of this background estimate depends on the occlusion level of the background in the images processed. Depending on the particular characteristics of the image sequence, our algorithm can recover successfully the template of the moving object when using the average of the images as the initial estimate of the background. This is the case with the image sequence we use in the experiment reported in section 4.3. In this section we propose a more elaborate initialization that leads to better initial estimates of the background.

Rather than ignoring the moving objects, we initialize the background estimate by averaging the registered images, excluding the regions detected as being in movement. It corresponds to taking in our algorithm as the initial estimate $\hat{\mathbf{T}}$ for the object template \mathbf{T} the detected moving regions. These regions do not necessarily form the exact object templates. We call these moving regions the object mask. The object mask is computed by averaging a set of two-frame mask estimates. Each two-frame mask estimate is obtained from the moving regions detected between the two frames.

Given a pair of images, registered according to the estimate of the background motion, $\mathcal{M}(\mathbf{p}_f)\mathbf{I}_f$ and $\mathcal{M}(\mathbf{p}_g)\mathbf{I}_g$, we detect the region whose motion differs from the background motion by thresholding the difference between the given images:

$$\mathbf{R}_{fg}(x, y) = \begin{cases} 1 & \text{if } |\mathcal{M}(\mathbf{p}_f)\mathbf{I}_f(x, y) - \mathcal{M}(\mathbf{p}_g)\mathbf{I}_g(x, y)| > \eta \\ 0 & \text{otherwise} \end{cases}. \quad (4.1)$$

This region contains the template of the moving object in both images $\mathcal{M}(\mathbf{p}_f)\mathbf{I}_f$ and $\mathcal{M}(\mathbf{p}_g)\mathbf{I}_g$. If the moving object is sufficiently textured, and its intensity level is not very similar to the background intensity level, the region \mathbf{R}_{fg} contains the union of the templates of that object positioned in each of the frames (see example shown in Figure 4.1).

We obtain an estimate of the template of the object by intersecting \mathbf{R}_{fg} with itself,

registered according to the estimate of the motion of the object between frames \mathbf{I}_f and \mathbf{I}_g ,

$$\mathbf{M}_{fg} = \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{R}_{fg} \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{R}_{fg}. \quad (4.2)$$

We call the estimate \mathbf{M}_{fg} a two-frame mask of the moving object. In the example of Figure 4.1, this mask successfully detects the moving object.

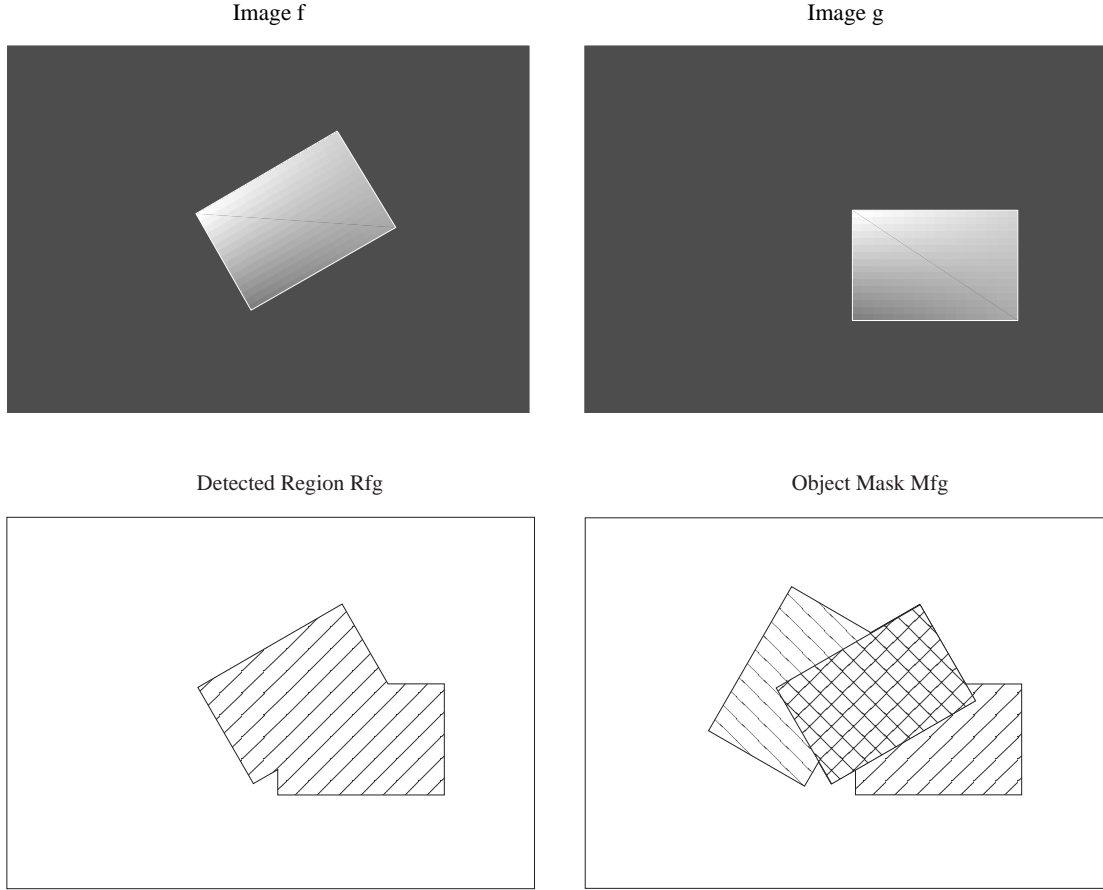


Figure 4.1: Object mask from a pair of images. On the top, images \mathbf{I}_f and \mathbf{I}_g . On the bottom left, the detected moving region \mathbf{R}_{fg} . On the bottom right, the two-frame object mask \mathbf{M}_{fg} obtained by intersecting \mathbf{R}_{fg} with itself, registered according to the motion of the object.

When the moving objects contain regions of low texture, or when its intensity level is very similar to the background intensity level, the detected moving region \mathbf{R}_{fg} does not detect the entire moving region, and \mathbf{M}_{fg} does not represent a reliable mask of the object. Also, with some specific object shape, or motions, or due to noise, the two-frame

mask \mathbf{M}_{fg} may contain regions not belonging to the object. To deal with this type of scenes, we take into account masks \mathbf{M}_{fg} obtained from several pairs of frames $\{(\mathbf{I}_f, \mathbf{I}_g)\}$, to generate the initial object template. We compute the average of these masks. This computation is done recursively by:

$$\overline{\mathbf{M}}_f = \frac{f-2}{f} \overline{\mathbf{M}}_{f-1} + \frac{2}{f(f-1)} \sum_{g=1}^{f-1} \mathbf{M}_{fg}. \quad (4.3)$$

We stop this recursion when $\overline{\mathbf{M}}_f$ stabilizes. Finally, the template of the object is initialized by thresholding $\overline{\mathbf{M}}_f$. This initial estimate of the template may not be a very accurate approximation to the shape of the moving object, but it proves to be good enough to initialize properly our segmentation algorithm.

4.3 Minimization Procedure: Two-Step Iterative Algorithm

In this section we develop the algorithm to segment a two-dimensional (2D) rigid moving object. Our algorithm is a feasible approximation of the *Maximum Likelihood* (ML) estimate of the parameters involved. For easy reference, we rewrite the ML cost function as derived in chapter 2,

$$C_2 = \iint \sum_{f=1}^F \left\{ \mathbf{I}_f(x, y) - \mathcal{M}(\mathbf{p}_f^\#) \mathbf{B}(x, y) \left[1 - \mathcal{M}(\mathbf{q}_f^\#) \mathbf{T}(x, y) \right] - \mathcal{M}(\mathbf{q}_f^\#) \mathbf{O}(x, y) \mathcal{M}(\mathbf{q}_f^\#) \mathbf{T}(x, y) \right\}^2 \mathbf{H}(x, y) dx dy, \quad (4.4)$$

where $\{\mathbf{I}_f, 1 \leq f \leq F\}$ is the image sequence, \mathbf{B} is the unknown texture of the background, to which we also call background world image, \mathbf{O} is the unknown texture of the moving object, or object world image, \mathbf{T} is the unknown template of the moving object, $\{\mathbf{p}_f, 1 \leq f \leq F\}$ represents the 2D motion of the background, $\{\mathbf{q}_f, 1 \leq f \leq F\}$ represents the 2D motion of the object, and \mathcal{M} is the motion operator introduced in section 2.2.

We estimate the position vectors $\{\mathbf{p}_f, \mathbf{q}_f, 1 \leq f \leq F\}$ on a frame by frame basis, as detailed in chapter 3. We introduce the estimates of the position vectors into the

ML cost function and minimize then with respect to the remaining unknowns, i.e., with respect to the template \mathbf{T} of the moving object, the texture \mathbf{O} of the moving object, and the texture \mathbf{B} of the background. To minimize the ML cost function, we propose a computationally simple two-step iterative algorithm.

Due to the special structure of the ML cost function C_2 , we can express explicitly and with no approximations involved the estimate $\hat{\mathbf{O}}$ of the object world image in terms of the template \mathbf{T} . Doing this, we are left with the minimization of C_2 with respect to the template \mathbf{T} and the background world image \mathbf{B} , still a nonlinear minimization. We approximate this minimization by a two-step iterative algorithm: (i) in step one, we solve for the background \mathbf{B} while the template \mathbf{T} is kept fixed; and (ii) in step two, we solve for the template \mathbf{T} while the background \mathbf{B} is kept fixed. We obtain closed-form solutions for the minimizers in each of the steps (i) and (ii). The two steps are repeated iteratively. The value of the ML cost function C_2 decreases along the iterative process. The algorithm proceeds till every pixel has been assigned unambiguously to either the moving object or to the background. Once the two-step iterative algorithm converges, i.e, when the template estimate $\hat{\mathbf{T}}$ stabilizes, if new frames are available, we update only the textures of the object and the background. The template estimate $\hat{\mathbf{T}}$ is frozen and, as new images become available, the object and background world images, $\hat{\mathbf{O}}$ and $\hat{\mathbf{B}}$, are updated by computing recursively their ML estimates. As it usually happens with iterative methods, the convergence to the global minimum of the ML cost function C_2 is not guaranteed, being necessary to provide a good initialization for the iterative algorithm. The initialization was considered in section 4.2. In our experiments, see section 4.4 and chapter 6, the algorithm converges in a small number of iterations, typically in three to five iterations.

Estimation of the moving object world image

We express the estimate $\hat{\mathbf{O}}$ of the moving object world image in terms of the object template \mathbf{T} . By minimizing C_2 with respect to the intensity value $\mathbf{O}(x, y)$, we obtain the

average of the pixels that correspond to the point (x, y) of the object. The estimate $\hat{\mathbf{O}}$ of the moving object world image is then

$$\hat{\mathbf{O}} = \mathbf{T} \frac{1}{F} \sum_{f=1}^F \mathcal{M}(\mathbf{q}_f) \mathbf{I}_f. \quad (4.5)$$

This compact expression averages the observations \mathbf{I} registered according to the motion \mathbf{q}_f of the object in the region corresponding to the template \mathbf{T} of the moving object.

We consider now separately the two steps of the iterative algorithm described above.

Step (i): estimation of the background for fixed template

To find the estimate $\hat{\mathbf{B}}$ of the background world image, given the template \mathbf{T} , we register each term of the sum of the ML cost function C_2 in equation (4.4) according to the position of the camera \mathbf{p}_f relative to the background. This is a valid operation because C_2 is defined as a sum over all the space $\{(x, y)\}$. We get

$$C_2 = \iint \sum_{f=1}^F \left\{ \mathcal{M}(\mathbf{p}_f) \mathbf{I}_f - \mathbf{B} \left[1 - \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \mathbf{T} \right] - \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \mathbf{O} \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \mathbf{T}(x, y) \right\}^2 \mathcal{M}(\mathbf{p}_f) \mathbf{H} \, dx \, dy. \quad (4.6)$$

Minimizing the ML cost function C_2 given by expression (4.6) with respect to the intensity value $\mathbf{B}(x, y)$, we get the estimate $\hat{\mathbf{B}}(x, y)$ as the average of the observed pixels that correspond to the pixel (x, y) of the background. The background world image estimate $\hat{\mathbf{B}}$ is then written as

$$\hat{\mathbf{B}} = \frac{\sum_{f=1}^F \left[1 - \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \mathbf{T} \right] \mathcal{M}(\mathbf{p}_f) \mathbf{I}_f}{\sum_{i=f}^F \left[1 - \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \mathbf{T} \right] \mathcal{M}(\mathbf{p}_f) \mathbf{H}}. \quad (4.7)$$

The estimate $\hat{\mathbf{B}}$ of the background world image in expression (4.7) is the average of the observations \mathbf{I}_f registered according to the background motion \mathbf{p}_i , in the regions $\{(x, y)\}$ not occluded by the moving object, i.e., when $\mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \mathbf{T}(x, y) = 0$. The term $\mathcal{M}(\mathbf{p}_f) \mathbf{H}$ provides the correct averaging normalization in the denominator by accounting only for the pixels seen in the corresponding image.

If we compare the moving object world image estimate $\hat{\mathbf{O}}$ given in equation (4.5) with the background world image estimate $\hat{\mathbf{B}}$ in equation (4.7), we see that $\hat{\mathbf{O}}$ is linear in the template \mathbf{T} , while $\hat{\mathbf{B}}$ is nonlinear in \mathbf{T} . This has implications when estimating the template \mathbf{T} of the moving object, as we see next.

Step (ii): estimation of the template for fixed background

Let the background world image \mathbf{B} be given and replace the object world image estimate $\hat{\mathbf{O}}$ given by expression (4.5) in expression (4.4). The ML cost function C_2 becomes linearly related to the object template \mathbf{T} . Manipulating C_2 as described next, we obtain

$$C_2 = \iint \mathbf{T}(x, y) \mathbf{Q}(x, y) dx dy + \text{Constant}, \quad (4.8)$$

$$\mathbf{Q}(x, y) = \mathbf{Q}_1(x, y) - \mathbf{Q}_2(x, y), \quad (4.9)$$

$$\mathbf{Q}_1(x, y) = \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} [\mathcal{M}(\mathbf{q}_f) \mathbf{I}_f(x, y) - \mathcal{M}(\mathbf{q}_g) \mathbf{I}_g(x, y)]^2, \quad (4.10)$$

$$\mathbf{Q}_2(x, y) = \sum_{f=1}^F \left[\mathcal{M}(\mathbf{q}_f) \mathbf{I}_f(x, y) - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y) \right]^2. \quad (4.11)$$

We call \mathbf{Q} the *segmentation matrix*. On the first reading, the reader may want to skip the derivation of expressions (4.8) to (4.11) and proceed after the symbol \square on page 78.

Derivation of expressions (4.8) to (4.11)

Replace the estimate $\hat{\mathbf{O}}$ of the moving object world image, given by expression (4.5), in expression (4.4), to obtain

$$C_2 = \iint \sum_{f=1}^F \left\{ \mathbf{I} - \mathcal{M}(\mathbf{p}_f^\#) \mathbf{B} \left[1 - \mathcal{M}(\mathbf{q}_f^\#) \mathbf{T} \right] - \frac{1}{F} \sum_{g=1}^F \mathcal{M}(\mathbf{q}_f^\# \mathbf{q}_g) \mathbf{I}_g \mathcal{M}(\mathbf{q}_f^\#) \mathbf{T} \right\}^2 \mathbf{H} dx dy. \quad (4.12)$$

Register each term of the sum according to the object position \mathbf{q}_f . This is valid because C_2 is defined as an integral over all the space $\{(x, y)\}$. The result is

$$C_2 = \iint \sum_{f=1}^F \left\{ \left[\mathcal{M}(\mathbf{q}_f) \mathbf{I}_f - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right] \right.$$

$$+ \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} - \frac{1}{F} \sum_{g=1}^F \mathcal{M}(\mathbf{q}_g) \mathbf{I}_g \right] \mathbf{T} \Big\}^2 \mathcal{M}(\mathbf{q}_f) \mathbf{H} \, dx \, dy. \quad (4.13)$$

In the remainder of the derivation, the spatial dependence is not important here, and we simplify the notation by omitting (x, y) . We rewrite the expression for C_2 in compact form as

$$C_2 = \iint \mathbf{C} \, dx \, dy, \quad \text{where} \quad \mathbf{C} = \sum_{f=1}^F \left\{ \left[\mathcal{I}_f - \mathcal{B}_f \right] + \left[\mathcal{B}_f - \frac{1}{F} \sum_{g=1}^F \mathcal{I}_g \right] \mathbf{T} \right\}^2 \mathcal{H}_f, \quad (4.14)$$

$$\mathcal{I}_f = \mathcal{M}(\mathbf{q}_f) \mathbf{I}_f(x, y), \quad \mathcal{B}_f = \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y), \quad \text{and} \quad \mathcal{H}_f = \mathcal{M}(\mathbf{q}_f) \mathbf{H}(x, y). \quad (4.15)$$

We need in the sequel the following equalities

$$\left[\sum_{g=1}^F \mathcal{I}_g \right]^2 = \sum_{f=1}^F \sum_{g=1}^F \mathcal{I}_f \mathcal{I}_g \quad \text{and} \quad \sum_{f=2}^F \sum_{g=1}^{f-1} [\mathcal{I}_f^2 + \mathcal{I}_g^2] = (F-1) \sum_{g=1}^F \mathcal{I}_g^2. \quad (4.16)$$

Manipulating \mathbf{C} under the assumption that the moving object is completely visible in the F images ($\mathbf{T} \mathcal{H}_f = \mathbf{T}, \forall_f$), and using the left equality in (4.16), we obtain

$$\mathbf{C} = \mathbf{T} \left\{ \sum_{f=1}^F [2\mathcal{I}_f \mathcal{B}_f - \mathcal{B}_f^2] - \frac{1}{F} \left[\sum_{g=1}^F \mathcal{I}_g \right]^2 \right\} + \sum_{f=1}^F [\mathcal{I}_f - \mathcal{B}_f]^2 \mathcal{H}_f. \quad (4.17)$$

The second term of \mathbf{C} in expression (4.17) is independent of the template \mathbf{T} . To show that the sum that multiplies \mathbf{T} is the segmentation matrix \mathbf{Q} as defined by expressions (4.9), (4.10), and (4.11), write \mathbf{Q} using the notation introduced in (4.15):

$$\mathbf{Q} = \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} [\mathcal{I}_f^2 + \mathcal{I}_g^2 - 2\mathcal{I}_f \mathcal{I}_g] - \sum_{f=1}^F [\mathcal{I}_f^2 + \mathcal{B}_f^2 - 2\mathcal{I}_f \mathcal{B}_f]. \quad (4.18)$$

Manipulating this equation, using the two equalities in (4.16), we obtain

$$\mathbf{Q} = \sum_{f=1}^F [2\mathcal{I}_f \mathcal{B}_f - \mathcal{B}_f^2] - \frac{1}{F} \left[\sum_{g=1}^F \mathcal{I}_g^2 + 2 \sum_{f=2}^F \sum_{g=1}^{f-1} \mathcal{I}_f \mathcal{I}_g \right]. \quad (4.19)$$

The following equality concludes the derivation:

$$\left[\sum_{g=1}^F \mathcal{I}_g \right]^2 = \sum_{g=1}^F \mathcal{I}_g^2 + 2 \sum_{f=2}^F \sum_{g=1}^{f-1} \mathcal{I}_f \mathcal{I}_g. \quad (4.20)$$

□

We estimate the template \mathbf{T} by minimizing the ML cost function given by expression (4.8) over the template \mathbf{T} , given the background world image \mathbf{B} . It is clear from expression (4.8), that the minimization of C_2 with respect to each spatial location of \mathbf{T} is independent from the minimization over the other locations. The template $\hat{\mathbf{T}}$ that minimizes the ML cost function C_2 is given by the following test evaluated at each pixel:

$$\mathbf{Q}_1(x, y) \begin{matrix} \hat{\mathbf{T}}(x, y) = 0 \\ \geq \\ \hat{\mathbf{T}}(x, y) = 1 \end{matrix} \mathbf{Q}_2(x, y). \quad (4.21)$$

The estimate $\hat{\mathbf{T}}$ of the template of the moving object in equation (4.21) is obtained by checking which of two accumulated square differences is greater. In the spatial locations where the accumulated differences between each frame $\mathcal{M}(\mathbf{q}_f)\mathbf{I}_f$ and the background $\mathcal{M}(\mathbf{q}_g\mathbf{p}_g^\#)\mathbf{B}$ are greater than the accumulated differences between each pair of co-registered frames $\mathcal{M}(\mathbf{q}_f)\mathbf{I}_f$ and $\mathcal{M}(\mathbf{q}_g)\mathbf{I}_g$, we estimate $\hat{\mathbf{T}}(x, y) = 1$, meaning that these pixels belong to the moving object. If not, the pixel is assigned to the background.

The reason why we did not replace the background world image estimate $\hat{\mathbf{B}}$ given by (4.7) in (4.4) as we did with the object world image estimate $\hat{\mathbf{O}}$ is that it leads to an expression for C_2 in which the minimization with respect to each different spatial location $\mathbf{T}(x, y)$ is not independent from the other locations. Solving this binary minimization problem by a conventional method is extremely time consuming. In contrast, the minimization of C_2 over \mathbf{T} for fixed \mathbf{B} results in a local binary test. This makes our solution computationally simple.

It may happen that, after processing the F available frames, the test (4.21) remains inconclusive at a given pixel (x, y) ($\mathbf{Q}_1(x, y) \simeq \mathbf{Q}_2(x, y)$): in other words, it is not possible to decide if this pixel belongs to the moving object or to the background. We modify our algorithm to address this ambiguity by defining the modified cost function

$$C_{2\text{MOD}} = C_2 + \alpha \text{Area}(\mathbf{T}) = C_2 + \alpha \iint \mathbf{T}(x, y) dx dy, \quad (4.22)$$

where C_2 is as in equation (4.8), α is non-negative, and $\text{Area}(\mathbf{T})$ is the area of the

template. Minimizing $C_{2\text{MOD}}$ balances the agreement between the observations and the model (term C_2), with minimizing the area of the template. Carrying out the minimization, first note that the second term in expression (4.22) does not depend on \mathbf{O} , neither on \mathbf{B} , so we get $\hat{\mathbf{O}}_{\text{MOD}} = \hat{\mathbf{O}}$ and $\hat{\mathbf{B}}_{\text{MOD}} = \hat{\mathbf{B}}$. By replacing $\hat{\mathbf{O}}$ in $C_{2\text{MOD}}$, we get a modified version of equation (4.8),

$$C_{2\text{MOD}} = \iint \mathbf{T}(x, y) [\mathbf{Q}(x, y) + \alpha] dx dy + \text{Constant}, \quad (4.23)$$

where \mathbf{Q} is defined in equations (4.9), (4.10), and (4.11). The template estimate is now given by the following test, that extends test (4.21),

$$\mathbf{Q}(x, y) \begin{cases} \hat{\mathbf{T}}(x, y) = 0 \\ \geq \\ \hat{\mathbf{T}}(x, y) = 1 \end{cases} - \alpha. \quad (4.24)$$

The parameter α may be chosen by experimentation, by using the *Minimum Description Length* (MDL) principle, see reference [12], or made adaptive by a annealing schedule like in stochastic relaxation.

Once the two-step iterative algorithm converges, i.e, when the template estimate $\hat{\mathbf{T}}$ stabilizes, if new frames are available, we proceed updating the world images. The template estimate $\hat{\mathbf{T}}$ is frozen and, as new images become available, the object and background world images, $\hat{\mathbf{O}}$ and $\hat{\mathbf{B}}$, are updated by computing recursively the ML estimate. The outputs of the two-step iterative method are the template estimate $\hat{\mathbf{T}}$, and world images estimates $\hat{\mathbf{O}}_i$ and $\hat{\mathbf{B}}_i$. We index the world image estimates by i to make it explicit that these estimates are after the i images processed by the two-step algorithm. These estimates result from equations (4.5) and (4.7), using the template estimate $\hat{\mathbf{T}}$.

Recursive generation of the world images

As new images are available, we update the estimates of the world images. To be computationally efficient, we rewrite equations (4.5) and (4.7) in recursive form. From (4.5), we obtain

$$\hat{\mathbf{O}}_f = \frac{f-1}{f} \hat{\mathbf{O}}_{f-1} + \frac{1}{f} \hat{\mathbf{T}} \mathcal{M}(\mathbf{q}_f) \mathbf{I}_f. \quad (4.25)$$

Before we present the recursive form of equation (4.7) for updating the estimate $\hat{\mathbf{B}}_f$, we introduce the background estimation weights \mathbf{S}_f that correspond to the denominator in equation (4.7),

$$\mathbf{S}_f = \sum_{g=1}^f \left[\mathbf{1} - \mathcal{M}(\mathbf{p}_g \mathbf{q}_g^\#) \hat{\mathbf{T}} \right] \mathcal{M}(\mathbf{p}_g) \mathbf{H}. \quad (4.26)$$

The element $\mathbf{S}_f(x, y)$ is the number of times the pixel $\mathbf{B}(x, y)$ has been observed in the first f frames. The matrix \mathbf{S}_f is written recursively as

$$\mathbf{S}_f = \mathbf{S}_{f-1} + \left[\mathbf{1} - \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \hat{\mathbf{T}} \right] \mathcal{M}(\mathbf{p}_f) \mathbf{H}. \quad (4.27)$$

We rewrite recursively the background world image estimate $\hat{\mathbf{B}}_f$ in equation (4.7) by expressing the sum in the numerator in terms of $\hat{\mathbf{B}}_{f-1}$, \mathbf{S}_{f-1} , and \mathbf{I}_f . We get¹

$$\hat{\mathbf{B}}_f = \frac{\mathbf{S}_{f-1}}{\mathbf{S}_f} \hat{\mathbf{B}}_{f-1} + \frac{\mathbf{1}}{\mathbf{S}_f} \left[\mathbf{1} - \mathcal{M}(\mathbf{p}_f \mathbf{q}_f^\#) \hat{\mathbf{T}} \right] \mathcal{M}(\mathbf{p}_f) \mathbf{I}_f. \quad (4.28)$$

4.4 Experiment

We apply the segmentation algorithm to one computer generated image sequence using real images. The experiment illustrates the convergence of the two-step iterative algorithm.

We synthesized an image sequence according to the *Generative Video* (GV) [34, 35, 36, 37] model described in chapter 2. Figure 4.2 shows the world images used. The left frame, from a real video, is the background world image. The moving object template is the logo of the *Instituto Superior Técnico* (IST) which is transparent between the letters. Its world image, shown in the right frame, is obtained by clipping with the IST logo a portion of one of the frames in the sequence. The task of reconstructing the object template is particularly challenging with this video sequence due to the low contrast between the object and the background and the complexity of the template. We synthesized a sequence of 20 images where the background is static and the IST logo moves around.

¹We assume in (4.28) that $\mathbf{S}_f(x, y) \neq 0$. If $\mathbf{S}_f(x, y) = 0$, the corresponding (x, y) entry of $\hat{\mathbf{B}}_f$ is zero.

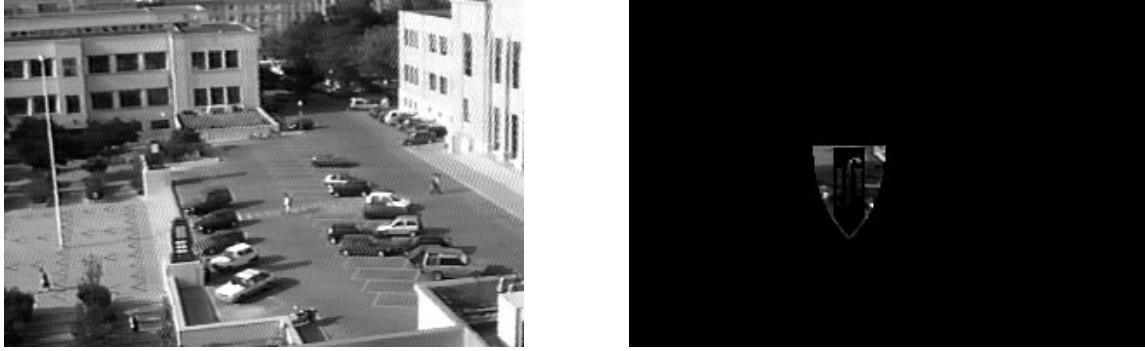


Figure 4.2: GV constructs: background and moving object.

Figure 4.3 shows three frames of the sequence obtained according to the image formation model introduced in chapter 2, expression (2.2), with noise variance $\sigma^2 = 4$ (the intensity values are in the interval $[0, 255]$). The object moves from the center (left frame) down by translational and rotational motion. It is difficult to recognize the logo in the two right frames because its texture is confused with the texture of the background.



Figure 4.3: Three frames of the GV synthesized image sequence.

Figure 4.4 illustrates the four iterations it took for the two-step estimation method of our algorithm to converge. The template estimate is initialized to zero (top left frame). Each background estimate in the right hand side was obtained using the template estimate on the left of it. Each template estimate was obtained using the previous background estimate. The arrows in Figure 4.4 indicate the flow of the algorithm. The good template estimate obtained, see bottom left image, illustrates that our algorithm can estimate complex templates in low contrast background.

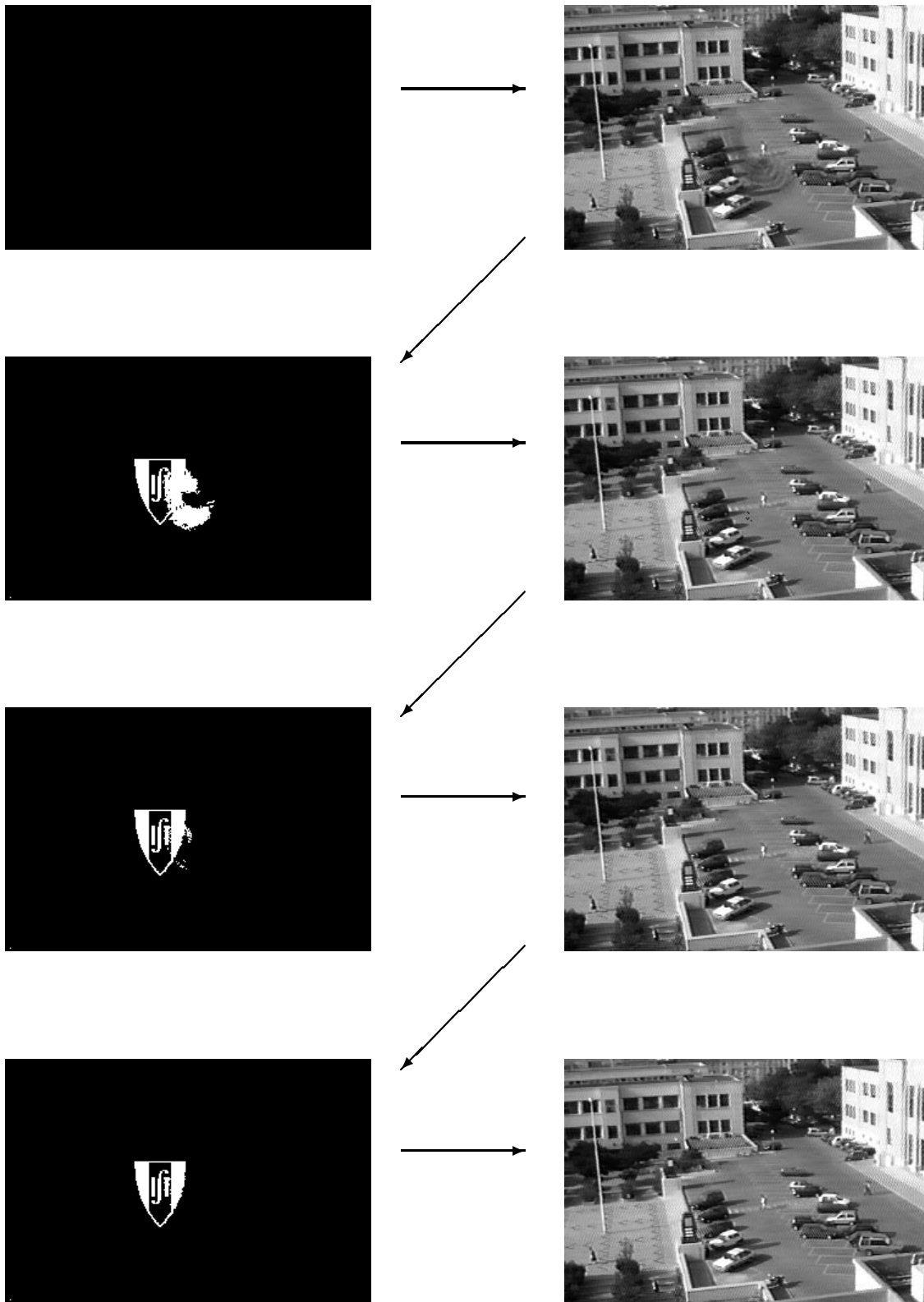


Figure 4.4: Two-step iterative method: template estimates and background estimates.

Note that this type of complex templates (objects with transparent regions) is much easier to describe by using a binary matrix than by using contour based descriptions, like splines, Fourier descriptors, or snakes. Our algorithm overcomes the difficulty arising from the higher number of degrees of freedom of the binary template by integrating over time the small intensity differences between the background and the object. The two-step iterative algorithm performs this integrations in an expedite way.

4.5 Summary

In this chapter, we develop an algorithm for segmenting a two-dimensional (2D) rigid moving object from an image sequence. Our method recovers the template of the 2D rigid moving object by processing directly the image intensity values. We model the rigidity of the moving object over a set of frames and motivate our algorithm by looking for a feasible approximation to the *Maximum Likelihood* (ML) estimation of the unknowns involved in the segmentation problem.

Our methodology introduces the 2D motion estimates into the ML cost function and uses a two-step iterative algorithm to approximate the minimization of the resultant cost function. The steps of the iterative algorithm are: (i) estimate the background for fixed template, (ii) estimate the template for fixed background. We start by describing a method to initialize the algorithm, i.e, a method to compute a first estimate of the object template. Then, we develop the two-step iterative algorithm. We obtain closed-form solutions for both steps (i) and (ii). The solutions for steps (i) and (ii) result computationally very simple. The two-step algorithm is computationally efficient because the convergence is achieved in a small number of iterations (typically three to five iterations).

The chapter ends with an illustrative experiment. We show that the algorithm proposed can recover complex templates in a low contrast scene.

Chapter 5

Analysis of the Segmentation Algorithm

5.1 Introduction

In chapter 4 we proposed a method to segment a two-dimensional (2D) rigid moving object. Our method approximates the *Maximum Likelihood* (ML) estimate of the unknowns involved in the segmentation problem. To minimize the ML cost function, we presented in chapter 4 a computationally simple two-step iterative algorithm. This chapter is devoted to the study of the behavior of the two-step algorithm.

The two steps of the iterative algorithm are the following: (i) estimation of the background texture assuming that the object template is known; (ii) estimation of the object template assuming that the background texture is known. In step (i), we estimate the background texture by averaging the appropriate regions of the observed frames. In step (ii), the estimate of the template of the moving object leads to a binary test evaluated at each pixel. After processing a small number of frames, it may happen that the background is correctly estimated but some of the template pixels are incorrectly classified, i.e., classified as belonging to the object template when they do not belong to the object template, or classified as not belonging to the object template when they belong to the object template. In fact, the background estimates are accurate because they are obtained by averaging the observed frames in step (i) of the algorithm. The background

estimates are robust to errors in the previous estimates of the template because the average smoothes out the intensity of any pixel that may be taken into account incorrectly. On the other hand, even if the background is correctly estimated, some of the template pixels may be incorrectly classified in step (ii) of the algorithm. This is because the binary test involved in step (ii) of the algorithm may provide incorrect pixel classifications if the difference of intensities between the object and the background is very small when compared to the observation noise.

In this chapter we develop an upper bound for the probability of misclassification of the pixels of the template by using the *Tchebycheff inequality* [46]. We illustrate the behavior predicted by the theoretical analysis through an experiment with simulated data. The chapter is organized as follows. In section 5.2 we derive expressions for the mean and variance of the *segmentation matrix* \mathbf{Q} involved in the binary test of step (ii) of the algorithm introduced in chapter 4. In section 5.3 we develop an upper bound for the probability of misclassifying the template pixels. Section 5.4 describes one experiment that confirms the theoretical analysis. Section 5.5 concludes the chapter.

5.2 Statistics of the Segmentation Matrix

The estimate $\hat{\mathbf{T}}$ of the template of the moving object is constructed from the test (4.21). We study the behavior of this test and the convergence of the template estimate $\hat{\mathbf{T}}$ as we process additional images in the sequence. For easy reference, we rewrite in compact form the *segmentation matrix* \mathbf{Q} involved in test (4.21),

$$\mathbf{Q} = \mathbf{Q}_1 - \mathbf{Q}_2, \quad (5.1)$$

$$\mathbf{Q}_1 = \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f) \mathbf{I}_f - \mathcal{M}(\mathbf{q}_g) \mathbf{I}_g \right]^2, \quad (5.2)$$

$$\mathbf{Q}_2 = \sum_{f=1}^F \left[\mathcal{M}(\mathbf{q}_f) \mathbf{I}_f - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \hat{\mathbf{B}} \right]^2, \quad (5.3)$$

where $\{\mathbf{I}_f\}$ is the image sequence and $\hat{\mathbf{B}}$ is the estimate of the background. The operator \mathcal{M} is the motion operator introduced in chapter 2, section 2.3.

To develop upper bounds for the probability of misclassification of the pixels of the template, we need to compute first the mean and variance of the segmentation matrix \mathbf{Q} given by expressions (5.1), (5.2), and (5.3). The following statistical analysis is conditioned on the estimate of the background being equal to the actual background, i.e., conditioned on $\hat{\mathbf{B}} = \mathbf{B}$. This approximation is supported from a practical viewpoint because, as mentioned in the previous section, the background estimate is robust to errors in the previous estimate of the template.

Mean of \mathbf{Q}

We compute successively the means of the matrices \mathbf{Q}_1 and \mathbf{Q}_2 given by expressions (5.2) and (5.3). We start with the mean of \mathbf{Q}_1 . We replace the model for the image sequence $\{\mathbf{I}_f\}$ given by expression (2.2) in expression (5.2) and compute the expected value of \mathbf{Q}_1 with respect to the observation noise.

Replacing the image sequence model (2.2) in expression (5.2) we get

$$\mathbf{Q}_1 = \frac{1}{N} \sum_{f=2}^F \sum_{g=1}^{f-1} \left\{ \left\{ \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} [\mathbf{1} - \mathbf{T}] + \mathbf{O} \mathbf{T} + \mathcal{M}(\mathbf{q}_f) \mathbf{W}_f \right\} \mathcal{M}(\mathbf{q}_f) \mathbf{H} - \left\{ \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} [\mathbf{1} - \mathbf{T}] + \mathbf{O} \mathbf{T} + \mathcal{M}(\mathbf{q}_g) \mathbf{W}_g \right\} \mathcal{M}(\mathbf{q}_g) \mathbf{H} \right\}^2. \quad (5.4)$$

Manipulating \mathbf{Q}_1 under the assumption that the moving object is completely visible in the F images ($\forall f : \mathbf{T} \mathcal{M}(\mathbf{q}_f) \mathbf{H} = \mathbf{T}$), we obtain

$$\mathbf{Q}_1 = \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left\{ \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \mathcal{M}(\mathbf{q}_f) \mathbf{H} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \mathcal{M}(\mathbf{q}_g) \mathbf{H} \right] [\mathbf{1} - \mathbf{T}] + \mathcal{M}(\mathbf{q}_f) \mathbf{W} \mathcal{M}(\mathbf{q}_f) \mathbf{H} - \mathcal{M}(\mathbf{q}_g) \mathbf{W} \mathcal{M}(\mathbf{q}_g) \mathbf{H} \right\}^2. \quad (5.5)$$

We now compute the expected value of \mathbf{Q}_1 in expression (5.5) with respect to the observation noise sequence $\{\mathbf{W}\}$. With zero mean white noise,

$$\mathbb{E}\{\mathbf{W}_f\} = \mathbf{0}, \quad (5.6)$$

$$\mathbb{E}\{\mathbf{W}_f \mathbf{W}_g\} = \sigma^2 \delta_{fg} \mathbf{1}. \quad (5.7)$$

The expected value of \mathbf{Q}_1 is, after expanding the square in expression (5.5) and using the expectations (5.6) and (5.7),

$$\begin{aligned} E\{\mathbf{Q}_1\} = & \left[\mathbf{1} - \mathbf{T} \right] \frac{1}{F} \sum_{f=2}^N \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \mathcal{M}(\mathbf{q}_f) \mathbf{H} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \mathcal{M}(\mathbf{q}_g) \mathbf{H} \right]^2 \\ & + \frac{\sigma^2}{F} \sum_{f=2}^N \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f) \mathbf{H} + \mathcal{M}(\mathbf{q}_g) \mathbf{H} \right]. \quad (5.8) \end{aligned}$$

We now compute the expected value of the matrix \mathbf{Q}_2 . Replacing the image sequence model (2.2) in expression (5.3), we get

$$\mathbf{Q}_2 = \sum_{f=1}^F \left\{ \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \left[\mathbf{1} - \mathbf{T} \right] + \mathbf{O} \mathbf{T} + \mathcal{M}(\mathbf{q}_f) \mathbf{W}_f \mathcal{M}(\mathbf{q}_f) \mathbf{H} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \hat{\mathbf{B}} \right\}^2. \quad (5.9)$$

Assuming that the background \mathbf{B} equals its estimate $\hat{\mathbf{B}}$ and that the object is completely visible in the F frames, i.e., $\forall f : \mathbf{T} \mathcal{M}(\mathbf{q}_f) \mathbf{H} = \mathbf{T}$, we obtain

$$\mathbf{Q}_2 = \sum_{f=1}^F \left\{ \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right] \mathbf{T} + \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \left[\mathcal{M}(\mathbf{q}_f) \mathbf{H} - \mathbf{1} \right] + \mathcal{M}(\mathbf{q}_f) \mathbf{W} \mathcal{M}(\mathbf{q}_f) \mathbf{H} \right\}^2. \quad (5.10)$$

We compute the expected value of \mathbf{Q}_2 conditioned on $\hat{\mathbf{B}} = \mathbf{B}$, which we denote by $E_B\{\mathbf{Q}_2\}$. Using the expectations (5.6) and (5.7), we obtain

$$E_B\{\mathbf{Q}_2\} = \mathbf{T} \sum_{f=1}^F \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right]^2 + \sum_{f=1}^F \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}^2 \left[\mathbf{1} - \mathcal{M}(\mathbf{q}_f) \mathbf{H} \right] + \sigma^2 \sum_{f=1}^F \mathcal{M}(\mathbf{q}_f) \mathbf{H}. \quad (5.11)$$

The expected value of the segmentation matrix \mathbf{Q} in expression (5.1) conditioned on $\hat{\mathbf{B}} = \mathbf{B}$ is now given by

$$E_B\{\mathbf{Q}\} = E\{\mathbf{Q}_1\} - E_B\{\mathbf{Q}_2\}. \quad (5.12)$$

For the regions $\{(x, y)\}$ that fall inside all the F images, we have

$$\forall f : \mathcal{M}(\mathbf{q}_f) \mathbf{H}(x, y) = 1. \quad (5.13)$$

For these regions, by replacing the results (5.8) and (5.11) in equation (5.12), we get for the expected value of the segmentation matrix \mathbf{Q} ,

$$\begin{aligned} E_B \{\mathbf{Q}\} = [\mathbf{1} - \mathbf{T}] \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \right]^2 \\ - \mathbf{T} \sum_{f=1}^F \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right]^2 - \sigma^2 \mathbf{1} \quad (5.14) \end{aligned}$$

To obtain (5.14), we used (5.13) in expressions (5.8) and (5.11).

The coefficients that multiply \mathbf{T} and $[\mathbf{1} - \mathbf{T}]$ in expression (5.14) are non-negative. Thus, as we process additional images, $E_B \{\mathbf{Q}(x, y)\}$ becomes more negative in the spatial locations where $\mathbf{T}(x, y) = 1$, and more positive where $\mathbf{T}(x, y) = 0$ (the term $\sigma^2 \mathbf{1}$ is negligible because its magnitude is in general much smaller than the magnitude of the first two terms). After processing a sufficiently large number of frames, the sign of $E_B \{\mathbf{Q}\}$ at each pixel (x, y) stabilizes. The estimate of the moving object template is given by the test (4.21), i.e., we estimate $\hat{\mathbf{T}}(x, y) = 1$ if $\mathbf{Q}(x, y) < 0$ and $\hat{\mathbf{T}}(x, y) = 0$ if $\mathbf{Q}(x, y) > 0$. The convergence of the mean value $E_B \{\mathbf{Q}(x, y)\}$ to a positive value where $\mathbf{T}(x, y) = 0$, and a negative value where $\mathbf{T}(x, y) = 1$, by itself, does not guarantee that the probability of misclassifying the template pixels is low. We must study the evolution of the variance of the segmentation matrix $\mathbf{Q}(x, y)$.

Variance of \mathbf{Q}

We now compute the variance of \mathbf{Q} conditioned on $\hat{\mathbf{B}} = \mathbf{B}$ denoted by $V_B \{\mathbf{Q}\}$. Using $E_B \{\mathbf{Q}\}$ given by expression (5.14), and the image sequence model given by expression (2.2), we compute the expected value $V_B \{\mathbf{Q}\}$ of the matrix $[\mathbf{Q} - E_B \{\mathbf{Q}\}]^2$ with respect to the observation noise,

$$V_B \{\mathbf{Q}\} = E_B \left\{ \left[\mathbf{Q} - E_B \{\mathbf{Q}\} \right]^2 \right\}. \quad (5.15)$$

To compute the conditional variance of the segmentation matrix \mathbf{Q} given $\hat{\mathbf{B}} = \mathbf{B}$, we first express the matrix $\mathbf{Q} - E_B \{\mathbf{Q}\}$ in terms of the background intensity levels \mathbf{B} , the

moving object intensity levels \mathbf{O} , the template \mathbf{T} , and the observation noise \mathbf{W} . From equations (5.5), (5.8), (5.10), (5.11), and using (5.13), we obtain

$$\begin{aligned}
\mathbf{Q} - \mathbb{E}_B \{ \mathbf{Q} \} &= \mathbf{Q}_1 - \mathbb{E} \{ \mathbf{Q}_1 \} - \mathbf{Q}_2 + \mathbb{E}_B \{ \mathbf{Q}_2 \} \\
&= \left[\mathbf{1} - \mathbf{T} \right] \frac{2}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \right] \left[\mathcal{M}(\mathbf{q}_f) \mathbf{W} - \mathcal{M}(\mathbf{q}_g) \mathbf{W} \right] \\
&\quad - \mathbf{T} 2 \sum_{f=1}^F \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right] \mathcal{M}(\mathbf{q}_f) \mathbf{W} \\
&\quad + \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f) \mathbf{W} - \mathcal{M}(\mathbf{q}_g) \mathbf{W} \right]^2 - \sum_{f=1}^F \mathcal{M}(\mathbf{q}_f) \mathbf{W}^2 + \sigma^2 \mathbf{1}. \quad (5.16)
\end{aligned}$$

To simplify the notation, we define the matrices \mathbf{P}_1 , as the coefficient of the term that multiplies $[\mathbf{1} - \mathbf{T}]$, \mathbf{P}_2 , as the coefficient of the term that multiplies \mathbf{T} , and \mathbf{P}_3 , as the remaining term in expression (5.16),

$$\mathbf{P}_1 = \frac{2}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \right] \left[\mathcal{M}(\mathbf{q}_f) \mathbf{W} - \mathcal{M}(\mathbf{q}_g) \mathbf{W} \right], \quad (5.17)$$

$$\mathbf{P}_2 = 2 \sum_{f=1}^F \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right] \mathcal{M}(\mathbf{q}_f) \mathbf{W}, \quad (5.18)$$

$$\mathbf{P}_3 = \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f) \mathbf{W} - \mathcal{M}(\mathbf{q}_g) \mathbf{W} \right]^2 - \sum_{f=1}^F \mathcal{M}(\mathbf{q}_f) \mathbf{W}^2 + \sigma^2 \mathbf{1}, \quad (5.19)$$

and write the matrix $[\mathbf{Q} - \mathbb{E}_B \{ \mathbf{Q} \}]^2$ as

$$\left[\mathbf{Q} - \mathbb{E}_B \{ \mathbf{Q} \} \right]^2 = \left[\mathbf{1} - \mathbf{T} \right] \mathbf{P}_1^2 + \mathbf{T} \mathbf{P}_2^2 + \mathbf{P}_3^2 + \left[\mathbf{1} - \mathbf{T} \right] \mathbf{P}_1 \mathbf{P}_3 - \mathbf{T} \mathbf{P}_2 \mathbf{P}_3. \quad (5.20)$$

We compute the expected value of the matrices in the right hand side of expression (5.20) with respect to the observation noise. Using the following expectations that result from the zero mean and whiteness of the noise and the properties of higher-order moments of Gaussian random variables,

$$\mathbb{E} \left\{ \mathbf{W}_f \mathbf{W}_g \mathbf{W}_h \right\} = \mathbf{0}, \quad (5.21)$$

$$\mathbb{E} \left\{ \mathbf{W}_f \mathbf{W}_g \mathbf{W}_h \mathbf{W}_i \right\} = \frac{\sigma^4}{3} \left[\delta_{fg} \delta_{hi} (1 + 2\delta_{fh}) + \delta_{fh} \delta_{gi} (1 + 2\delta_{fg}) + \delta_{fi} \delta_{gh} (1 + 2\delta_{fg}) \right] \mathbf{1}, \quad (5.22)$$

and the equalities

$$\sum_{i=1}^N i = \frac{N(N+1)}{2} \quad \text{and} \quad \sum_{i=1}^N i^2 = \frac{N(N+1)(2N+1)}{6}, \quad (5.23)$$

we obtain for the expected values of \mathbf{P}_1^2 , \mathbf{P}_2^2 , \mathbf{P}_3^2 , $\mathbf{P}_1\mathbf{P}_3$, and $\mathbf{P}_2\mathbf{P}_3$,

$$\mathbb{E} \{ \mathbf{P}_1^2 \} = \frac{4\sigma^2}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \right]^2, \quad (5.24)$$

$$\mathbb{E} \{ \mathbf{P}_2^2 \} = 4\sigma^2 \sum_{f=1}^F \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right]^2, \quad (5.25)$$

$$\mathbb{E} \{ \mathbf{P}_3^2 \} = 2\sigma^4 \mathbf{1}, \quad \mathbb{E} \{ \mathbf{P}_1 \mathbf{P}_3 \} = \mathbf{0}, \quad \text{and} \quad \mathbb{E} \{ \mathbf{P}_2 \mathbf{P}_3 \} = \mathbf{0}. \quad (5.26)$$

Using these results, we compute the expected value of the matrix $[\mathbf{Q} - \mathbb{E}_B \{ \mathbf{Q} \}]^2$ given by expression (5.20). We obtain for the variance $V_B \{ \mathbf{Q} \} = \mathbb{E} \{ [\mathbf{Q} - \mathbb{E}_B \{ \mathbf{Q} \}]^2 \}$,

$$\begin{aligned} V_B \{ \mathbf{Q} \} = & \left[\mathbf{1} - \mathbf{T} \right] \frac{4\sigma^2}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B} \right]^2 \\ & + \mathbf{T} 4\sigma^2 \sum_{f=1}^F \left[\mathbf{O} - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B} \right]^2 + 2\sigma^4 \mathbf{1}. \end{aligned} \quad (5.27)$$

From expression (5.27), we see that the variance of the segmentation matrix \mathbf{Q} increases as we process more frames. This is because when more frames are taken into account, matrix \mathbf{Q} , given by expressions (5.1), (5.2), and (5.3), sums a larger number of random variables. We now relate the behavior of the mean and variance of the segmentation matrix \mathbf{Q} with the estimate of the moving object template.

5.3 Bounds on the Probability of Error

We develop an upper bound for the probability of misclassification of the template pixels. The estimate of the template of the moving object is given by test (4.21). For easy reference, we rewrite the test. The estimate $\hat{\mathbf{T}}$ is

$$\mathbf{Q}_1(x, y) \begin{matrix} \hat{\mathbf{T}}(x, y) = 0 \\ \geq \\ \hat{\mathbf{T}}(x, y) = 1 \end{matrix} \mathbf{Q}_2(x, y) \iff \mathbf{Q}(x, y) \begin{matrix} \hat{\mathbf{T}}(x, y) = 0 \\ \geq \\ \hat{\mathbf{T}}(x, y) = 1 \end{matrix} 0, \quad (5.28)$$

where \mathbf{Q}_1 , \mathbf{Q}_2 , and \mathbf{Q} are given by expressions (5.2), (5.3), and (5.1).

We upper bound the probability of misclassification of a pixel as belonging or not to the moving object template by using the *Tchebycheff inequality* [46]. We use equations (5.14) and (5.27) for the mean and variance of the segmentation matrix \mathbf{Q} and expression (5.28) for the classification of the template pixels. We derive upper bounds on the probability of the two types of error: type-I error – misclassifying a pixel as not belonging to the template when it actually belongs to the template (probability of a miss); type-II error – misclassifying a pixel as belonging to the template when it actually does not belong to the template (probability of false alarm).

Type-I error

If the pixel (x, y) belongs to the template of the moving object, we have $\mathbf{T}(x, y) = 1$. The mean $E_B \{\mathbf{Q}\}$ and the variance $V_B \{\mathbf{Q}\}$ are given by making $\mathbf{T} = \mathbf{1}$ in expressions (5.14) and (5.27). We get

$$E_B \{\mathbf{Q}(x, y)\} = - \sum_{f=1}^F \left[\mathbf{O}(x, y) - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y) \right]^2 - \sigma^2, \quad (5.29)$$

$$V_B \{\mathbf{Q}(x, y)\} = 4\sigma^2 \sum_{f=1}^F \left[\mathbf{O}(x, y) - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y) \right]^2 + 2\sigma^4. \quad (5.30)$$

The probability that the pixel (x, y) is misclassified as not belonging to the template (type-I error) is given by

$$P_B \left\{ \mathbf{T}(x, y) = 1 \wedge \widehat{\mathbf{T}}(x, y) = 0 \right\} = P_B \{ \mathbf{Q}(x, y) > 0 \} \quad (5.31)$$

(see test (5.28)). Using elementary properties of the absolute value, the right hand side of equality (5.31) is majored as

$$P_B \{ \mathbf{Q}(x, y) > 0 \} \leq P_B \left\{ \left| \mathbf{Q}(x, y) - E_B \{ \mathbf{Q}(x, y) \} \right| > -E_B \{ \mathbf{Q}(x, y) \} \right\}. \quad (5.32)$$

To upper-bound the probability in the right hand side of expression (5.32) in terms of the mean and variance of \mathbf{Q} , we use the Tchebycheff inequality [46], a result of probability

theory that states that for any random variable x with mean η and variance σ^2 and for any $\epsilon > 0$, the probability that x takes values outside the interval $[\eta - \epsilon, \eta + \epsilon]$ is bounded by

$$P\{|x - \eta| > \epsilon\} \leq \frac{\sigma^2}{\epsilon^2}. \quad (5.33)$$

By using the Tchebycheff inequality (5.33) we upper-bound the probability in the right hand side of expression (5.32) as

$$P_B\left\{\left|\mathbf{Q}(x, y) - E_B\{\mathbf{Q}(x, y)\}\right| > -E_B\{\mathbf{Q}(x, y)\}\right\} \leq \frac{V_B\{\mathbf{Q}(x, y)\}}{\left[-E_B\{\mathbf{Q}(x, y)\}\right]^2}. \quad (5.34)$$

By replacing expressions (5.29) and (5.30) into the right hand side of expression (5.34), we obtain the final upper-bound for the probability of the type-I error,

$$P_B\left\{\mathbf{T}(x, y) = 1 \wedge \hat{\mathbf{T}}(x, y) = 0\right\} \leq \frac{4\sigma^2}{\sum_{f=1}^F \left[\mathbf{O}(x, y) - \mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y)\right]^2} \quad (5.35)$$

(we simplified the expression of the bound (5.35) by neglecting the terms σ^2 and $2\sigma^4$ in expressions (5.29) and (5.30))

The bound on the probability of error given by expression (5.35) is inversely proportional to the square of the difference between the background intensity level and the moving object intensity level. This is in agreement with the intuition that the more the intensity levels of the background are different from the intensity level of the moving object, the less probable it is to misclassify an object pixel as belonging to the background. As we process additional images, the value of the error upper-bound given by expression (5.35) decreases and so the template estimates become more accurate with higher probability.

Type-II error

In a similar way, we find an upper bound for the probability of the type-II error. If the pixel (x, y) does not belong to the template, we have $\mathbf{T}(x, y) = 0$. The mean $E_B\{\mathbf{Q}\}$ and variance $V_B\{\mathbf{Q}\}$ are given by making $\mathbf{T} = \mathbf{0}$ in expressions (5.14) and (5.27). We get

$$E_B\{\mathbf{Q}(x, y)\} = \frac{1}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y) - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B}(x, y) \right]^2 - \sigma^2, \quad (5.36)$$

$$V_B \{ \mathbf{Q}(x, y) \} = \frac{4\sigma^2}{F} \sum_{f=2}^F \sum_{g=1}^{f-1} \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y) - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B}(x, y) \right]^2 + 2\sigma^4. \quad (5.37)$$

The upper-bound to the probability that the pixel (x, y) is misclassified as belonging to the template (type-II error) is successively given by

$$\begin{aligned} P_B \left\{ \mathbf{T}(x, y) = 0 \wedge \hat{\mathbf{T}}(x, y) = 1 \right\} &= P_B \{ \mathbf{Q}(x, y) < 0 \} \\ &\leq P_B \left\{ \left| \mathbf{Q}(x, y) - E_B \{ \mathbf{Q}(x, y) \} \right| > E_B \{ \mathbf{Q}(x, y) \} \right\} \\ &\leq \frac{E_B \{ \mathbf{Q}(x, y) \}}{\left[E_B \{ \mathbf{Q}(x, y) \} \right]^2} \\ &= \frac{4\sigma^2}{\frac{1}{F} \sum_f \sum_g \left[\mathcal{M}(\mathbf{q}_f \mathbf{p}_f^\#) \mathbf{B}(x, y) - \mathcal{M}(\mathbf{q}_g \mathbf{p}_g^\#) \mathbf{B}(x, y) \right]^2}. \end{aligned} \quad (5.38)$$

The first equality is from test (5.28). The first inequality derives from the properties of the absolute value. The second inequality is from the Tchebycheff inequality (5.33) and the last equality is from expressions (5.29) and (5.30), after neglecting the terms σ^2 and $2\sigma^4$.

The bound on the probability of error given by expression (5.38) is inversely proportional to the square of the difference between the background intensity levels registered according to the moving object positions. If different regions of the background have similar intensity levels and their relative location is such that they could be interpreted as belonging to the moving object, then the denominator in the bound (5.38) is small and the bound is high. When additional images are processed, these differences accumulate, the denominator in (5.38) increases, the value of the bound on the probability of error decreases, and the template estimates are more accurate with higher probability.

Note that for both pixels belonging to the template, or for pixels corresponding to the background, the probability of error is bounded by the inverse of the absolute value of $E_B \{ \mathbf{Q} \}$ at each pixel (x, y) . As we process additional images, the absolute value of $E_B \{ \mathbf{Q} \}$ increases (see expression (5.14)), the probability of error decreases, the sign of \mathbf{Q} at each pixel (x, y) stabilizes, which in turn, through test (5.28), stabilizes the estimate $\hat{\mathbf{T}}$ of the template. This temporal integration of the information in the video sequence is a

distinguishing feature of our approach. We succeed in segmenting low textured objects because the moving object template is estimated from the differences between the intensity levels of the object and the background, rather than from the motion of the brightness pattern (which is impossible to compute in low textured regions). Since our algorithm accumulates these differences across the time, we also succeed in segmenting objects that move against low contrast background.

5.4 Experiment

We describe one experiment that illustrates the behavior of the algorithm proposed in chapter 4, whose convergence was studied in sections 5.2 and 5.3.

We illustrate the template estimation step for a sequence of one-dimensional (1D) frames obtained with the *Generative Video* (GV) building blocks of Figure 2.1. We synthesized an image sequence by using the model in expression (2.2). The camera position was chosen constant and the object position was set to increase linearly with time. The frame sequence obtained is represented in Figure 5.1. Time increases from bottom to top. From the plot of Figure 5.1 we can see that the background is stationary and the object moves from the left to the right.

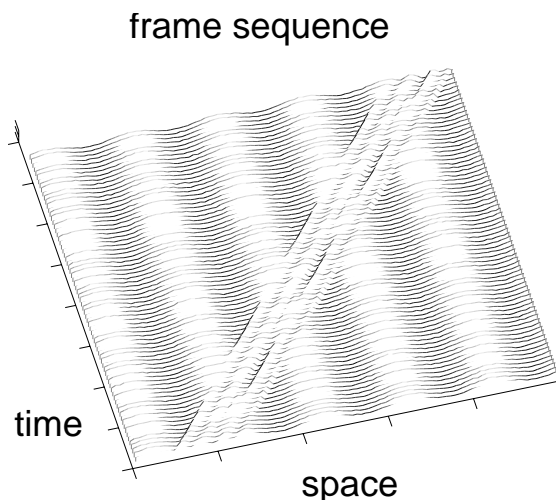


Figure 5.1: 1D image sequence synthesized with the GV constructs of Figure 2.1. Time increases from bottom to top.

The evolutions of matrices \mathbf{Q}_1 and \mathbf{Q}_2 (in this experiment, \mathbf{Q}_1 and \mathbf{Q}_2 are vectors because the frames are 1D) are represented by plots in Figure 5.2. The left plot represents the evolution of \mathbf{Q}_1 , while the right plot represents \mathbf{Q}_2 . Time increases from bottom to top. At the beginning, when only a few frames have been taken into account, the values of \mathbf{Q}_1 and \mathbf{Q}_2 are small and the test (5.28) is inconclusive. As more observations are processed, the absolute value of the difference between \mathbf{Q}_1 and \mathbf{Q}_2 rises and the test becomes unambiguous, see the evolution of the segmentation matrix $\mathbf{Q} = \mathbf{Q}_1 - \mathbf{Q}_2$ shown in the plot of Figure 5.3. When enough frames were processed, \mathbf{Q} takes high positive values for pixels that do not belong to the template of the moving object, and negative values for pixels belonging to the template, see the shape of \mathbf{Q} in the top of Figure 5.3 (the straight line at the bottom represents $\mathbf{Q} = \mathbf{0}$) and the template of the moving object in Figure 5.2.

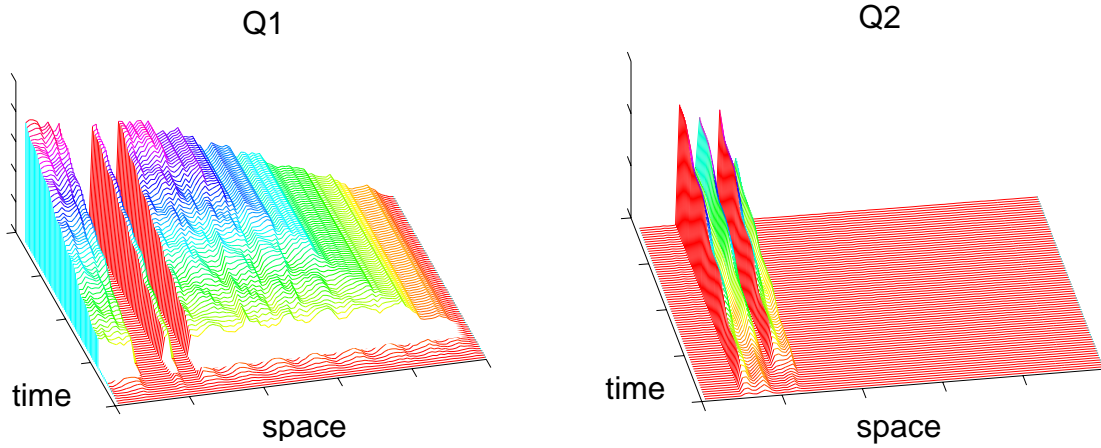


Figure 5.2: Evolution of \mathbf{Q}_1 and \mathbf{Q}_2 for the image sequence of Figure 5.1. Time increases from bottom to top.

On the plot of Figure 5.4 we show a grey level representation of the evolution of the result of the test (5.28). Time increases from bottom to top. Regions classified as belonging to the object template are light. Regions classified as not belonging to the template are dark. Middle grey regions correspond to the test (5.28) being inconclusive. Note that, after processing a number of frames, the regions are either light or dark, meaning that the

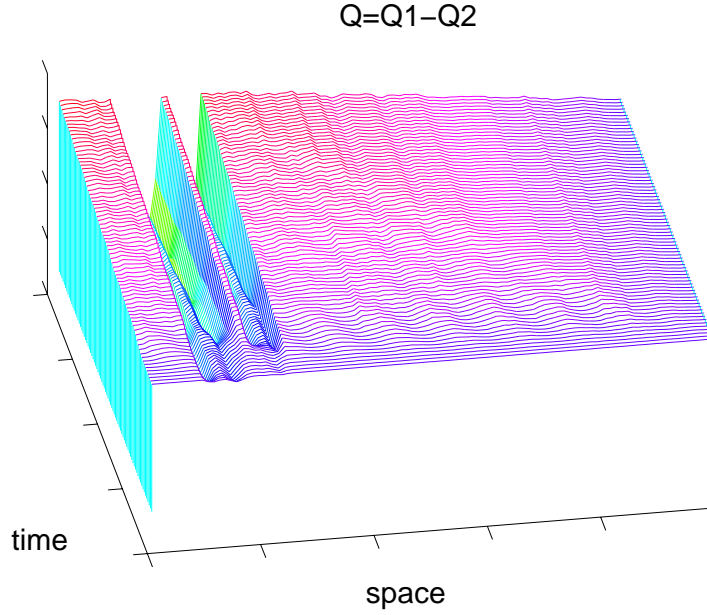


Figure 5.3: Evolution of the segmentation matrix \mathbf{Q} for the image sequence of Figure 5.1. Time increases from bottom to top.

test (5.28) is unambiguous at every spatial location. Figure 5.4 illustrates the convergent behavior of the template test. It is in agreement with the analysis made in the two previous sections. The estimates of the template of the moving object in Figure 5.4 confirm the statement above about the evolution of the segmentation matrix \mathbf{Q} in Figure 5.3, i.e., we see that the sequence of estimates of the template converges to the true template, represented in Figure 2.1.

The top of the plot in Figure 5.4 shows the final estimate of the template of the moving object. It is equal to the actual template, represented in Figure 2.1. In this example, the template of the moving object is the union of two disjoint intervals. We see that the segmentation algorithm recovers successfully the template of the moving object even when it is a disconnected set of pixels.

5.5 Conclusions

This chapter studies the behavior of the binary test involved in the two-step iterative algorithm proposed in chapter 4. The statistical analysis shows that the probability of

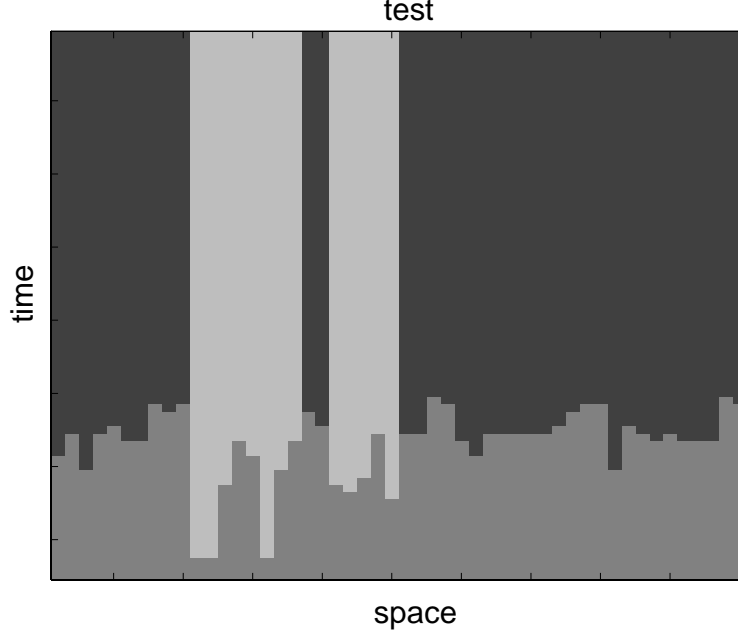


Figure 5.4: 1D template estimation for the sequence of Figure 5.1. Time increases from bottom to top. Regions classified as belonging to the object template are light. Regions classified as not belonging to the template are dark. Middle grey regions correspond to the test (5.28) being inconclusive.

misclassifying the pixels of the template of the moving object decreases as the number of frames processed increases. We illustrated this convergent behavior with an experiment that segments a moving object composed by two disjoint regions.

We develop upper bounds for two probability of errors when classifying pixels as belonging or not to the template of the moving object: type-I error – misclassifying a pixel as not belonging to the template when it actually belongs; and type-II error – misclassifying a pixel as belonging to the template when it actually does not belong. We start by computing the mean and variance of the *segmentation matrix* \mathbf{Q} involved in the template estimation test. To develop the upper bounds on the probability of each type of error, we use the *Tchebycheff inequality*.

We show that the upper bound on the probability of type-I error is inversely proportional to the square of the difference between the background intensity level and the moving object intensity level. This is in agreement with the intuition that the more the

intensity levels of the background differ from the intensity level of the moving object, the less probable it is to misclassify an object pixel as belonging to the background. As we process additional images, the value of the error bound decreases and the template estimates are more accurate.

The bound on the probability of type-II error is inversely proportional to the square of the difference between the background intensity levels registered according to the moving object positions. This agrees with the intuition that the uncertainty is higher when different regions of the background have similar intensity levels and their relative location is such that they could be interpreted as belonging to the moving object. When additional images are processed, these differences accumulate and the accuracy of the template estimates increases.

We illustrate the convergent behavior of the template test with an experiment with synthetic data. We use a one-dimensional image sequence that shows a moving object composed by two disjoint intervals. The time evolution of the segmentation matrix \mathbf{Q} is in agreement with the statistical analysis made in the chapter, i.e., \mathbf{Q} has values close to zero for all pixels, when a few frames are processed, and, as additional frames are processed, the absolute value of the entries of \mathbf{Q} rises. After processing a number of frames, the estimate of the template of the moving object is equal to the true template. The experiment illustrates that our segmentation algorithm recovers successfully templates that are disconnected sets of pixels.

Chapter 6

Real Video Experiments

In this chapter we present two experiments using real life video sequences. The first experiment illustrates the time evolution of the estimate of the moving object template when segmenting a robot soccer scene. The second experiment constructs the *Generative Video* (GV) [34, 35, 36, 37] representation of a real life traffic video clip.

6.1 Robot Soccer

We used a sequence of 20 images obtained from a robot soccer game, see references [54, 66]. It shows a white robot pursuing the ball. Frames 1, 4, 8, and 16 of the robot soccer video sequence are in Figure 6.1.

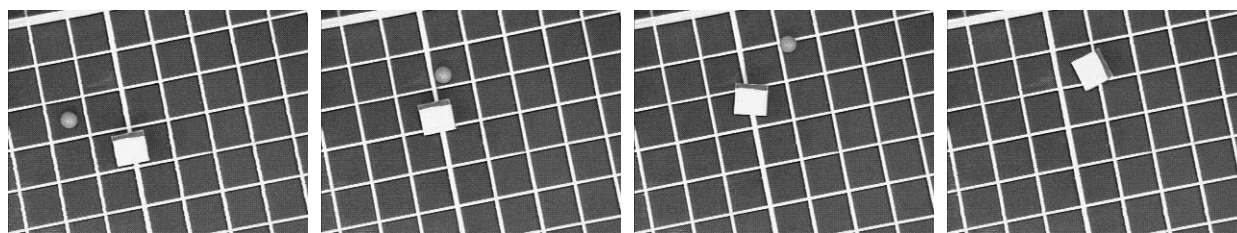


Figure 6.1: Robot soccer image sequence. Frames 1, 4, 8, and 16.

Although it is an easy task for humans to segment correctly the video sequence in Figure 6.1, even looking at a single frame, this is not the case when motion is the only cue taken into account. In fact, due to the low texture of the regions of both the field and the robot, the robot template is ambiguous during the first frames of the sequence. This

is because several regions belonging to the field can be incorrectly classified as belonging to the robot, since the motion of the robot during the first frames is such that the video sequence would be the same whether or not those regions move rigidly with the robot. The same happens to regions of the robot that can be interpreted as being stationary with respect to the field. Only after the robot rotates, it is possible to determine, without ambiguity, its template.

We applied the proposed segmentation algorithm to the video sequence of Figure 6.1. The initialization phase, described in section 4.2, took 5 frames. Applying the moving object template test, in expression (4.21), see section 4.3, the ball template becomes unambiguous after the 5 frames. Figure 6.2 shows the evolution of the robot template. Regions where the test is inconclusive are grey, regions classified as being part of the robot template are white, and regions classified as being part of the background are black. The robot template is unambiguous after 10 frames. The final robot template estimate is shown on the right side of Figure 6.2.



Figure 6.2: Estimate of robot template after frames 2, 4, 6, and 10.

Figure 6.3 illustrates the evolution of the segmentation matrix \mathbf{Q} introduced in section 4.3. The curves on the left side plot represent the value of $\mathbf{Q}(x, y)$ for representative pixels (x, y) in the template of the robot. These curves start close to zero and decrease with the number of frames processed, as predicted by the analysis in chapter 5. The curves on the right side plot of Figure 6.3 represent the evolution of $\mathbf{Q}(x, y)$ for pixels not in the template of the robot. For these pixels, $\mathbf{Q}(x, y)$ increases with the number of frames, again according to the analysis in chapter 5. Thus, while during the first frames the value

of $\mathbf{Q}(x, y)$ is close to zero and the template test is ambiguous (due to the low texture of the scene), after processing enough images the absolute value of $\mathbf{Q}(x, y)$ increases and the robot template becomes unambiguous.

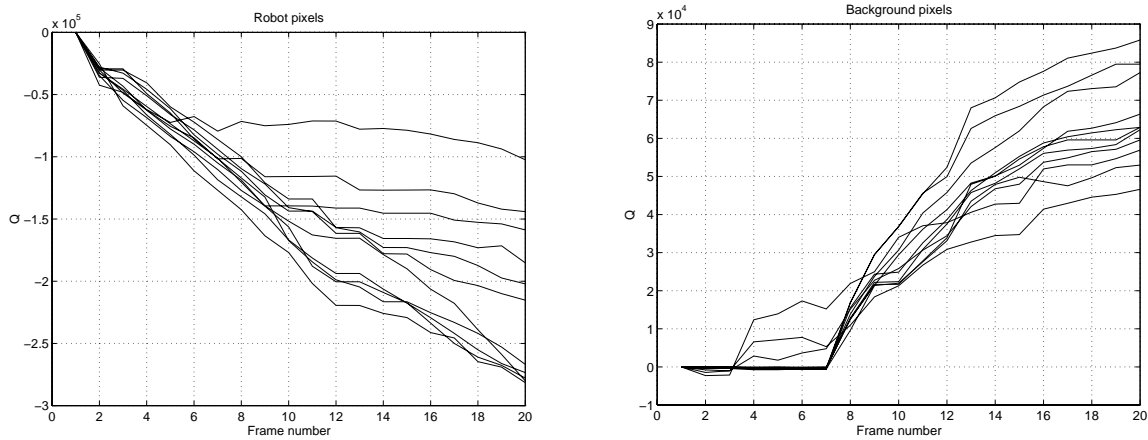


Figure 6.3: Evolution of the entries $\mathbf{Q}(x, y)$ of the segmentation matrix \mathbf{Q} for representative pixels: left plots are for pixels (x, y) in the robot template; right plots are for pixels (x, y) not in the robot template.

Figure 6.4 shows the recovered world images for the two moving objects and background, after processing the entire sequence of 20 frames.

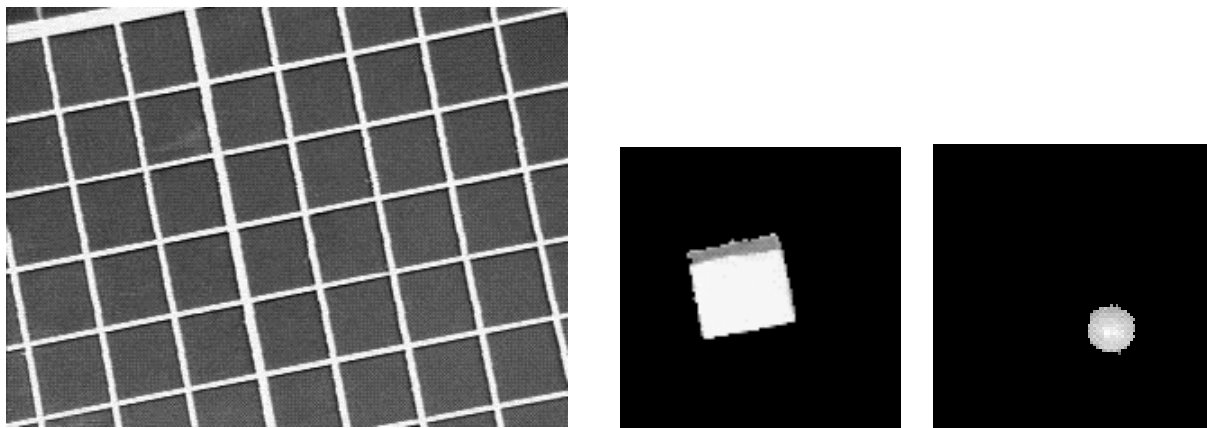


Figure 6.4: Background, robot, and ball world images recovered from the robot soccer video sequence of Figure 6.1.

6.2 Road Traffic

In this experiment we use a road traffic video clip. The road traffic video sequence has 250 frames. Figure 6.5 shows frames 15, 48, 166, and 225. The example given in chapter 1 to motivate the study of the segmentation of low textured scenes, see Figure 1.1, section 1.1, also uses frames 76 and 77 from the road traffic video clip.



Figure 6.5: Traffic road video sequence. Frames 15, 48, 166, and 225.

In this video sequence, the camera exhibits a pronounced panning motion, while four different cars enter and leave the scene. The cars and the background have regions of low texture. The intensity of some of the cars is very similar to the intensity of parts of the background. The example in chapter 5, section 5.4, illustrating the convergence of the estimate of the template, see Figures 5.1 through 5.4, is inspired in this type of scenes.

Figures 6.6 and 6.7 show the good results obtained after segmenting the sequence with our algorithm. Figure 6.7 displays the background world image, while Figure 6.6 shows the world images of each of the moving cars. The estimates of the templates for the cars in Figure 6.6 becomes unambiguous after 10, 12, 10, and 14 frames, respectively.

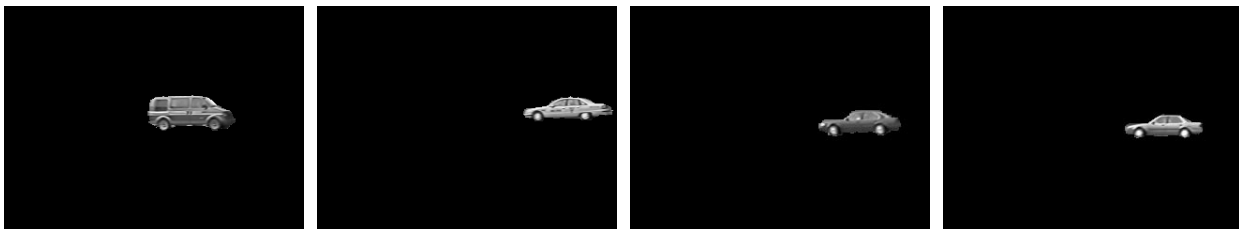


Figure 6.6: Moving objects world images recovered from the traffic road video sequence.



Figure 6.7: Background world image recovered from the traffic road video sequence.

Part II

Inference of 3D Rigid Structure

Chapter 7

3D Structure from 2D Video

7.1 Introduction

In Part II of the thesis we study the problem of inferring three-dimensional (3D) rigid structure from two-dimensional (2D) video by using the analogy with a communication system, as introduced in chapter 1, section 1.2, and illustrated in Figure 1.2. We address the recovery of 3D structure through the *Maximum Likelihood* (ML) estimation of the unknowns involved in the problem – the 3D shape, the 3D motions, and the texture. The formulation of the ML estimate from a set of frames leads to the minimization of a complex cost function. We do not attempt the minimization of the ML cost function with respect to the entire set of parameters by using generic optimization methods. Rather, we exploit the specific characteristics of the problem to develop a computationally feasible approximation to the ML solution.

In this chapter we state the problem of inferring 3D structure from video and describe our approach. The chapter is organized in the following way. In section 7.2 we contrast our approach with related research work. Section 7.3 formulates the problem by using the analogy with a classical communication system. In section 7.4 we discuss the ML estimate. We show that the ML estimate of the texture is obtained in terms of the unknown 3D shape and 3D motion. Section 7.5 describes our approach to the minimization of the ML cost function. We show that the classic *structure from motion* (SFM) approach is an approximation to the ML estimate. Section 7.6 summarizes the content of the chapter.

7.2 Related Work

The automatic generation of a three-dimensional (3D) description of the real world environment has received the attention of a large number of researchers. Target applications for this kind of models are found in several fields that go well beyond digital video. The information source for a number of successful approaches to 3D object modeling has been the range image, see for example references [22, 42, 50, 52]. This image, obtained from a range sensor, provides the distance between the sensor and the environment in front of it, on a uniform discrete grid. Since the range image itself contains explicit information about the 3D structure of the environment, the above cited works deal with the problem of how to combine a number of sets of 3D points (each set corresponding to a range image) into a 3D model. These approaches differ on the 3D model used, either a surface-based description [50, 52], or a discretized volumetric model [22, 42].

In this thesis we address the problem of building a 3D model from video data, when no explicit 3D information is given. The recovery of the 3D structure (3D shape and 3D motion) of rigid objects from a two-dimensional (2D) video sequence has been widely considered by the computer vision community. Methods that infer 3D shape from a single frame are based on cues such as shading and defocus. These methods fail to give reliable 3D shape estimates for unconstrained real-world scenes. If no prior knowledge about the scene is available, the cue to estimating the 3D structure is the 2D motion of the brightness pattern in the image plane. For this reason, this problem is generally referred to as *structure from motion* (SFM). The two major steps in SFM are usually the following: compute the 2D motion in the image plane; and estimate the 3D shape and the 3D motion from the computed 2D motion.

Early approaches to SFM processed a single pair of consecutive frames and provided existence and uniqueness results to the problem of estimating 3D motion and absolute depth from the 2D motion in the camera plane between two frames, see for example references [63, 64]. Two-frame based algorithms are highly sensitive to image noise, and,

when the object is far from the camera, i.e., at a large distance when compared to the object depth, they fail even at low level image noise.

More recent research has been oriented towards the use of longer image sequences. For example, references [51, 58] use a Kalman filter to integrate along time a set of two-frame depth estimates, and references [17, 55] use nonlinear optimization to solve for the rigid 3D motion and the set of 3D positions of feature points tracked across a set of frames.

Tomasi and Kanade [59, 60, 61] introduced the *factorization method*, an elegant method to recover rigid structure from an image sequence. Instead of representing the 3D positions of feature points by their image coordinates and their depths, they adopt Ullman's original formulation of the SFM problem, see reference [65]. In references [59, 61], as in reference [65], the 3D positions of the feature points are expressed in terms of cartesian coordinates in a world-centered coordinate system. In the factorization method, the 2D projection of each feature point is tracked over the image sequence. The 3D shape and motion are then estimated by factorizing a measurement matrix whose entries are the set of trajectories of the feature point projections. Tomasi and Kanade pioneered the use of linear subspace constraints in motion analysis. In fact, the key idea underlying the factorization method is the fact that the rigidity of the scene imposes that the measurement matrix lives in a low dimensional subspace of the universe of matrices. Tomasi and Kanade have shown that the measurement matrix is a rank 3 matrix in a noiseless situation. References [59, 60, 61] use the orthographic projection model. The factorization method was later extended to the scaled-orthography and para-perspective models, see references [47, 48, 49], and to the multibody scenario, see references [19, 20, 21].

In Part II of the thesis, we use the same type of subspace constraints to solve SFM in the more general scenario of recovering 3D motion and a parameteric description of the 3D shape from a sequence of 2D motion parameters. Exploiting further the linear subspace constraints, we will see that the SFM problem is solved by factorizing a matrix that is rank 1 in a noiseless situation, rather than a rank 3 matrix as in the original factorization method.

The factorization method as developed by Tomasi and Kanade relies on the matching of a set of features along the image sequence. This task is difficult when processing noisy videos. In general, only distinguished points, as brightness corners, are used as “trackable” feature points. As a consequence, the approach of Tomasi and Kanade does not provide dense depth estimates. Under our more general scenario, rather than describing the 3D shape by the set of 3D positions of the feature points, we parameterize the shape of the object surface and show that this parameterization induces a parametric model for the 2D motion of the brightness pattern in the image plane. Instead of tracking pointwise features, we track larger regions where the image motion is described by a single set of parameters. To recover in an expedite way the 3D motion and 3D shape parameters from the image motion parameters, we introduce the *surface-based factorization*, a generalization of the original factorization method. Another relevant feature of our method concerns its computational simplicity. By making an appropriate linear subspace projection, we find the unknown 3D structure by factorizing a matrix that is rank 1 in a noiseless situation, rather than a rank 3 matrix as in the original factorization method of Tomasi and Kanade. This allows the use of faster iterative algorithms to compute the matrix that best approximates the data.

We propose a final step that refines the estimate of the 3D shape given by the surface-based factorization method by computing the 3D shape directly from the image intensity values through *Maximum Likelihood* (ML) estimation. To overcome the difficulties in estimating 3D structure through the 2D motion induced onto the image plane, some researchers have used techniques that infer 3D structure directly from the image intensity values. Horn and Weldon estimate directly the 3D structure parameters by using the *brightness change constraint* between two consecutive frames, see reference [29]. Heel builds on this work by using a *Kalman filter* to update the estimates over time, see reference [28]. We address the recovery of 3D structure from a video sequence by using the analogy with the communications system, as presented in chapter 1. The ML estimation of the unknowns involved leads to the minimization of a cost function expressed in terms

of image intensities. For this reason, we also address the problem of inferring 3D structure directly from the image intensity values, as references [28, 29] do. Our approach is distinct from the approach of reference [28] because we model the rigidity of the scene over the set of frames, instead of trying to fuse a set of possibly inaccurate estimates obtained from pairs of consecutive frames.

7.3 Problem Formulation

Our approach to the problem of inferring three-dimensional (3D) structure from a video sequence is based on the analogy between the image analysis task and a classical communication system, as introduced in chapter 1, section 1.2, and illustrated in Figure 1.2. The first step in formulating the problem according to this analogy is the definition of an observation model, i.e., a model for the images in the sequence.

We consider a rigid object \mathcal{O} moving in front of a camera. The object \mathcal{O} is described by its 3D shape \mathcal{S} and texture \mathcal{T} . The texture \mathcal{T} represents the light received by the camera after reflecting on the object surface, i.e., the texture \mathcal{T} is the object brightness as perceived by the camera. The texture depends on the object surface photometric properties, as well as on the environment illumination conditions. We assume that the texture does not change with time. The 3D shape \mathcal{S} is a representation of the surface of the object. We do not need to specify at this point what type of surface representation \mathcal{S} is. The derivations in this chapter are valid for both parametric or non-parametric representations of the 3D shape, e.g., a dense depth map.

To represent the 3D motion, we attach a coordinate system to the object and to the camera. We define the 3D motion of the object by specifying the position of the object coordinate system (o.c.s.) relative to the camera coordinate system (c.c.s.). The position and orientation of the object \mathcal{O} at time instant f is represented by a vector \mathbf{m}_f . This vector codes a rotation-translation pair that takes values in the group of the rigid transformations of the space, the special Euclidean group $SE(3)$. The 3D structure ob-

tained by applying the 3D rigid transformation coded by the vector \mathbf{m}_f to the object \mathcal{O} is represented by $\mathcal{M}(\mathbf{m}_f)\mathcal{O}$.

The frame \mathbf{I}_f , captured at time f , is modeled as a noisy observation of the projection of the object

$$\mathbf{I}_f = \mathcal{P}\left\{\mathcal{M}(\mathbf{m}_f)\mathcal{O}\right\} + \mathbf{W}_f. \quad (7.1)$$

For simplicity, the observation noise \mathbf{W}_f is zero mean, white, and Gaussian. We assume that \mathcal{P} is the orthogonal projection operator that is known to be a good approximation to the perspective projection when the relative depth of the scene is small when compared to the distance to the camera. The factorization algorithms presented in chapters 8 and 9 are derived from the orthogonal projection model. They can be extended to the scaled-orthography and the para-perspective models in the same way as references [47, 48, 49] do for the original factorization methods. Note, however, that all derivations in the current chapter carry over to the general perspective projection model without a single modification. The idea that motivates the algorithm proposed in chapter 10 is also valid for the perspective projection and the algorithm itself is easily modified to cope with this general projection model.

The operator \mathcal{P} returns the texture \mathcal{T} as a real valued function defined over the image plane. This function is a nonlinear mapping that depends on the object shape \mathcal{S} and the object position \mathbf{m}_f . The intensity level of the projection of the object at pixel \mathbf{u} on the image plane is

$$\mathcal{P}\left\{\mathcal{M}(\mathbf{m}_f)\mathcal{O}\right\}(\mathbf{u}) = \mathcal{T}(\mathbf{s}_f(\mathcal{S}, \mathbf{m}_f; \mathbf{u})), \quad (7.2)$$

where $\mathbf{s}_f(\mathcal{S}, \mathbf{m}_f; \mathbf{u})$ is the nonlinear mapping that lifts the point \mathbf{u} on the image \mathbf{I}_f to the corresponding point on the 3D object surface. This mapping $\mathbf{s}_f(\mathcal{S}, \mathbf{m}_f; \mathbf{u})$ is determined by the object shape \mathcal{S} , and the position \mathbf{m}_f . To simplify the notation, we will usually write explicitly only the dependence on f , i.e., $\mathbf{s}_f(\mathbf{u})$.

Figure 7.1 illustrates the lifting mapping $\mathbf{s}_f(\mathbf{u})$ and the direct mapping $\mathbf{u}_f(\mathbf{s})$ for the orthogonal projection of a two-dimensional object. The inverse mapping $\mathbf{u}_f(\mathbf{s})$ also

depends on \mathcal{S} and \mathbf{m}_f , but we will, again, usually show only explicitly the dependence on f . On the left of Figure 7.1, the point \mathbf{s} on the surface of the object projects onto $\mathbf{u}_f(\mathbf{s})$ on the image plane. On the right, pixel \mathbf{u} on the image plane is lifted to $\mathbf{s}_f(\mathbf{u})$ on the object surface. We assume that the object does not occlude itself, i.e., we have $\mathbf{u}_f(\mathbf{s}_f(\mathbf{u})) = \mathbf{u}$ and $\mathbf{s}_f(\mathbf{u}_f(\mathbf{s})) = \mathbf{s}$. The mapping $\mathbf{u}_f(\mathbf{s})$, seen as a function of the frame index f , for a particular surface point \mathbf{s} , is the trajectory of the projection of that point on the image plane, i.e., it is the motion induced on the image plane.

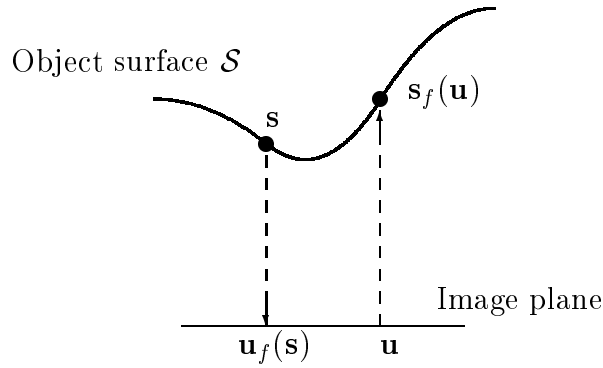


Figure 7.1: Mappings $\mathbf{u}_f(\mathbf{s})$ and $\mathbf{s}_f(\mathbf{u})$.

The observation model (7.1) is rewritten in terms of the object texture \mathcal{T} and the mappings $\mathbf{s}_f(\mathbf{u})$, by using the equality (7.2), as

$$\mathbf{I}_f(\mathbf{u}) = \mathcal{T}(\mathbf{s}_f(\mathbf{u})) + \mathbf{W}_f(\mathbf{u}). \quad (7.3)$$

Again, the dependence of \mathbf{u}_f on \mathcal{S} and \mathbf{m}_f is omitted for simplicity.

We consider the problem of recovering the real world constructs from the video sequence, i.e., recovering the 3D shape \mathcal{S} , the texture \mathcal{T} , and the 3D motion $\{\mathbf{m}_f, 1 \leq f \leq F\}$ of the object \mathcal{O} given the video sequence $\{\mathbf{I}_f, 1 \leq f \leq F\}$ of F frames.

7.4 Maximum Likelihood Estimation

Given the observation model, we recover the 3D shape, the texture, and the 3D motion of the object \mathcal{O} from the video sequence $\{\mathbf{I}_f, 1 \leq f \leq F\}$ by estimating the corre-

sponding unknowns: the 3D shape \mathcal{S} ; the texture \mathcal{T} ; and the set of 3D positions of the object $\{\mathbf{m}_f, 1 \leq f \leq F\}$ with respect to the camera.

We formulate the *Maximum Likelihood* (ML) solution. Assuming that the noise sequence $\{\mathbf{W}_f\}$ is zero mean, spatially and temporally white, and Gaussian, the ML estimate minimizes the sum over all the frames of the integral over the image plane of the squared errors between the observations and the model¹,

$$C_3(\mathcal{S}, \mathcal{T}, \{\mathbf{m}_f\}) = \sum_{f=1}^F \int [\mathbf{I}_f(\mathbf{u}) - \mathcal{T}(\mathbf{s}_f(\mathbf{u}))]^2 d\mathbf{u}, \quad (7.4)$$

$$\{\hat{\mathcal{S}}, \hat{\mathcal{T}}, \{\hat{\mathbf{m}}_f\}\} = \arg \min_{\mathcal{S}, \mathcal{T}, \{\mathbf{m}_f\}} C_3(\mathcal{S}, \mathcal{T}, \{\mathbf{m}_f\}). \quad (7.5)$$

In expression (7.4), we make explicit the dependence of the ML cost function C_3 on the object texture \mathcal{T} . Note that C_3 depends upon the object shape \mathcal{S} and the object positions $\{\mathbf{m}_f\}$ through the mappings $\{\mathbf{s}_f(\mathbf{u})\}$.

We address the minimization of the ML cost function $C_3(\mathcal{S}, \mathcal{T}, \{\mathbf{m}_f\})$ by first solving for the texture estimate $\hat{\mathcal{T}}$ in terms of the 3D object shape \mathcal{S} and the object positions $\{\mathbf{m}_f\}$.

We rewrite the ML cost function C_3 given by (7.4) by changing the integration variable from the image plane coordinate \mathbf{u} to the object surface coordinate \mathbf{s} . We obtain

$$C_3(\mathcal{S}, \mathcal{T}, \{\mathbf{m}_f\}) = \sum_{f=1}^F \int [\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathcal{T}(\mathbf{s})]^2 J_f(\mathbf{s}) d\mathbf{s}, \quad (7.6)$$

where $\mathbf{u}_f(\mathbf{s})$ is the mapping introduced above that projects the point \mathbf{s} on the object surface onto the image plane at instant f , see Figure 7.1. The function $J_f(\mathbf{s})$ is the Jacobian of the mapping $\mathbf{u}_f(\mathbf{s})$, $J_f(\mathbf{s}) = |\nabla \mathbf{u}_f(\mathbf{s})|$. Expression (7.6) shows that the ML cost function C_3 is quadratic in each intensity value $\mathcal{T}(\mathbf{s})$ of the object texture.

The ML estimate $\hat{\mathcal{T}}(\mathbf{s})$ is the function $\mathcal{T}(\mathbf{s})$ that minimizes C_3 given by expression (7.6). We show below that the ML estimate $\hat{\mathcal{T}}(\mathbf{s})$ of the texture $\mathcal{T}(\mathbf{s})$ is written in terms of the

¹We use a continuous spatial dependence for commodity. The variables \mathbf{u} and \mathbf{s} are continuous while f is discrete.

unknown 3D shape and 3D motion, through the mappings $\{\mathbf{u}_f(\mathbf{s})\}$, as

$$\hat{\mathcal{T}}(\mathbf{s}) = \frac{\sum_{f=1}^F \mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) J_f(\mathbf{s})}{\sum_{f=1}^F J_f(\mathbf{s})}. \quad (7.7)$$

Expression (7.7) states that the estimate of the texture of the object at the surface point \mathbf{s} is a weighted average of the measures of the intensity level corresponding to that surface point. At frame \mathbf{I}_f , a given region around \mathbf{s} on the object surface projects to a region around $\mathbf{u}_f(\mathbf{s})$. The size of this projected region changes with time because of the object motion. The more parallel to the image plane the tangent to the object surface at point \mathbf{s} is, the larger the size of the projected region is. Expression (7.7) shows that the larger the Jacobian $J_f(\mathbf{s})$ is, i.e., the larger the magnification of the region around \mathbf{s} is at frame \mathbf{I}_f , the larger is the weight given to that frame when estimating the texture $\mathcal{T}(\mathbf{s})$.

On first reading, the reader may want to skip the derivation of expression (7.7) and proceed after the symbol \square on page 118.

Derivation of expression (7.7)

To prove that the ML estimate $\hat{\mathcal{T}}(\mathbf{s})$ of the texture $\mathcal{T}(\mathbf{s})$ is given by expression (7.7), we show that it leads to the minimum of the cost function C_3 , given by expression (7.6), over all texture functions $\mathcal{T}(\mathbf{s})$.

Consider the candidate $\mathcal{T}(\mathbf{s}) = \hat{\mathcal{T}}(\mathbf{s}) + \mathcal{U}(\mathbf{s})$. The functional C_3 for the texture function $\mathcal{T}(\mathbf{s})$ is

$$\begin{aligned} C_3(\mathcal{T}) &= \sum_{f=1}^F \int \left[\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \hat{\mathcal{T}}(\mathbf{s}) - \mathcal{U}(\mathbf{s}) \right]^2 J_f(\mathbf{s}) d\mathbf{s} \\ &= \sum_{f=1}^F \int \left[\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \hat{\mathcal{T}}(\mathbf{s}) \right]^2 J_f(\mathbf{s}) d\mathbf{s} + \sum_{f=1}^F \int \mathcal{U}^2(\mathbf{s}) J_f(\mathbf{s}) d\mathbf{s} \\ &\quad - 2 \sum_{f=1}^F \int \left[\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \hat{\mathcal{T}}(\mathbf{s}) \right] \mathcal{U}(\mathbf{s}) J_f(\mathbf{s}) d\mathbf{s}. \end{aligned} \quad (7.8)$$

The first term of the expression above is $C_3(\hat{\mathcal{T}})$. The third term is 0, as a result of replacing $\hat{\mathcal{T}}(\mathbf{s})$ by expression (7.7). We now argue that the middle term is nonnegative.

If this is the case, then

$$C_3(\mathcal{T}) = C_3(\widehat{\mathcal{T}}) + \sum_{f=1}^F \int \mathcal{U}^2(\mathbf{s}) J_f(\mathbf{s}) d\mathbf{s} \geq C_3(\widehat{\mathcal{T}}), \quad (7.9)$$

which concludes the proof.

The inequality in expression (7.9) follows because we can always choose the texture coordinates \mathbf{s} in such a way that the determinants $J_f(\mathbf{s}) = |\nabla \mathbf{u}_f(\mathbf{s})|$ are positive. For example, make the texture coordinate \mathbf{s} equal to the image plane coordinate \mathbf{u} in the first frame \mathbf{I}_1 . The mapping $\mathbf{u}_1(\mathbf{s})$ is the identity mapping $\mathbf{u}_1(\mathbf{s}) = \mathbf{s}$ and we have a positive Jacobian $J_1(\mathbf{s}) = 1$. Now, draw an oriented closed contour on the surface \mathcal{S} , in the neighborhood of \mathbf{s} , and containing \mathbf{s} in its interior. This contour, which we call \mathcal{C}_s is projected in image \mathbf{I}_1 in an oriented closed planar contour \mathcal{C}_{u_1} . It is easy to see geometrically that in image \mathbf{I}_f the same contour \mathcal{C}_s projects to a contour \mathcal{C}_{u_f} that has, in general, different shape but the same orientation of the contour \mathcal{C}_{u_1} (recall that we are assuming that the object does not occlude itself). For this reason, the Jacobian $J_f(\mathbf{s})$ of the function that maps from \mathbf{s} to $\mathbf{u}_f(\mathbf{s})$ for $2 \leq f \leq F$ has the same sign as the Jacobian $J_1(\mathbf{s})$ of the function that maps from \mathbf{s} to $\mathbf{u}_1(\mathbf{s})$, so we get $J_f(\mathbf{s}) > 0$ for $1 \leq f \leq F$.

□

7.5 Structure from Motion: Approximate Maximum Likelihood Estimate

By inserting the texture estimate $\widehat{\mathcal{T}}$ given by expression (7.7) in expression (7.6), we express the *Maximum Likelihood* (ML) cost function C_3 in terms of the mappings $\{\mathbf{u}_f(\mathbf{s})\}$. After manipulations described below, we get

$$C_3(\mathcal{S}, \{\mathbf{m}_f\}) = \sum_{f=2}^F \sum_{g=1}^{f-1} \int \left[\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_g(\mathbf{u}_g(\mathbf{s})) \right]^2 \frac{J_f(\mathbf{s}) J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})} d\mathbf{s}. \quad (7.10)$$

The ML cost function C_3 in expression (7.10) is a weighted sum of the squared differences between all pairs of frames. At each surface point \mathbf{s} , the frame pair $\{\mathbf{I}_f, \mathbf{I}_g\}$ is weighted

by $\frac{J_f(\mathbf{s})J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})}$. The larger this weight is, i.e., the larger the magnification of a region around \mathbf{s} is in frames \mathbf{I}_f and \mathbf{I}_g , the more the square difference between \mathbf{I}_f and \mathbf{I}_g affects C_3 .

Expression (7.10) makes clear why the problem we are addressing is referred to as *structure from motion* (SFM): having eliminated the dependence on the texture, we are left with a cost function that depends on the *structure* (3D shape \mathcal{S} and 3D motion $\{\mathbf{m}_f\}$) only through the *motion* induced on the image plane, i.e., through the mappings $\{\mathbf{u}_f(\mathbf{s})\}$. Recall the comment on section 7.3 that $\mathbf{u}_f(\mathcal{S}, \mathbf{m}_f; \mathbf{s})$ depends on the shape \mathcal{S} and the motion \mathbf{m}_f .

On first reading, the reader may want to skip the derivation of expression (7.10) and proceed after the symbol \square on page 120.

Derivation of expression (7.10)

We show that the ML cost function is expressed in terms of the mappings $\{\mathbf{u}_f(\mathbf{s})\}$ as in expression (7.10).

Replace the texture estimate $\hat{\mathcal{T}}(\mathbf{s})$, given by expression (7.7), into the ML cost function, given by expression (7.6),

$$C_3 = \sum_{f=1}^F \int \left[\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \frac{\sum_{g=1}^F \mathbf{I}_g(\mathbf{u}_g(\mathbf{s})) J_g(\mathbf{s})}{\sum_{g=1}^F J_g(\mathbf{s})} \right]^2 J_f(\mathbf{s}) d\mathbf{s}. \quad (7.11)$$

Interchanging the sum and the integral in expression (7.11), we get, after simple algebraic manipulations,

$$C_3 = \int \sum_{f=1}^F \left[\frac{\sum_{g=1}^F [\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_g(\mathbf{u}_g(\mathbf{s}))] J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})} \right]^2 J_f(\mathbf{s}) d\mathbf{s}. \quad (7.12)$$

Expressing the square above in terms of a sum of products, we get

$$C_3 = \int \frac{\sum_{f=1}^F \sum_{g=1}^F \sum_{h=1}^F [\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_g(\mathbf{u}_g(\mathbf{s}))] [\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_h(\mathbf{u}_h(\mathbf{s}))] J_f(\mathbf{s}) J_g(\mathbf{s}) J_h(\mathbf{s})}{\left[\sum_{h=1}^F J_h(\mathbf{s}) \right]^2} d\mathbf{s}. \quad (7.13)$$

Carrying out the products and cancelling the symmetric terms of the sum, we get

$$C_3 = \int \frac{\sum_{f=1}^F \sum_{g=1}^F \sum_{h=1}^F [\mathbf{I}_f^2(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) \mathbf{I}_g(\mathbf{u}_g(\mathbf{s}))] J_f(\mathbf{s}) J_g(\mathbf{s}) J_h(\mathbf{s})}{\left[\sum_{h=1}^F J_h(\mathbf{s}) \right]^2} d\mathbf{s}. \quad (7.14)$$

In the integrand, we can cancel the factor $\sum_{h=1}^F J_h(\mathbf{s})$ in the numerator with one of the factors in the denominator, obtaining

$$C_3 = \int \frac{\sum_{f=1}^F \sum_{g=1}^F [\mathbf{I}_f^2(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_f(\mathbf{u}_f(\mathbf{s}))\mathbf{I}_g(\mathbf{u}_g(\mathbf{s}))] J_f(\mathbf{s})J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})} d\mathbf{s}. \quad (7.15)$$

By using the equality

$$\sum_{f=1}^F \sum_{g=1}^F [\mathbf{I}_f^2 - \mathbf{I}_f\mathbf{I}_g] J_f J_g = \sum_{f=2}^F \sum_{g=1}^{f-1} [\mathbf{I}_f - \mathbf{I}_g]^2 J_f J_g, \quad (7.16)$$

which is obtained by carrying out the square in the right hand side and rearranging the terms of the sum, we get

$$C_3 = \int \frac{\sum_{f=2}^F \sum_{g=1}^{f-1} [\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_g(\mathbf{u}_g(\mathbf{s}))]^2 J_f(\mathbf{s})J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})} d\mathbf{s}, \quad (7.17)$$

and conclude the derivation. Note that by interchanging the integral and the sum in expression (7.17), we get the ML cost function C_3 as in expression (7.10).

□

The SFM strategy can be understood as an approximation to the minimization of the functional (7.10) in two steps. The first step estimates the image motion, i.e., the mappings $\mathbf{u}_f(\mathbf{s})$. The second step estimates the shape \mathcal{S} and the 3D motion $\{\mathbf{m}_f\}$ from the image motion.

Our approach to the minimization of the ML cost function is the following. We compute the 3D motion by inferring SFM. Then we introduce the 3D motion estimates into the ML cost function and minimize it with respect to the unknown 3D shape. The rationale behind this approach is that the 3D motion can be inferred from the image motion computed at a sparse set of points or regions. The 3D shape is recovered from SFM based on a set of depths of points or regions for which the image motion was estimated. To refine the estimate of the 3D shape we infer the 3D shape directly from the image intensity values by introducing the 3D motion estimates into the ML cost function and minimizing it with respect to the unknown 3D shape.

Rigid structure from motion: surface-based rank 1 factorization

Chapters 8 and 9 show how to recover 3D rigid SFM. We start by estimating the image motion, i.e., the mappings $\mathbf{u}_f(\mathbf{s})$. This step is accomplished by minimizing an approximation to the ML cost function, as described in chapter 3. Then we estimate the 3D structure from the image motion measurements by solving the 3D shape \mathcal{S} and the 3D motion \mathbf{m}_f from the estimates of the image motion $\mathbf{u}_f(\mathbf{s})$.

We represent the 3D shape of the scene by a parametric description of the object surface. This representation induces a parametric model for the motion of the brightness pattern in the image plane. To recover simultaneously the parameters describing the 3D shape and the parameters describing the 3D motion from the parameters that describe the image motion, we develop an efficient method. The rigidity of the scene and the orthogonal projection model impose a specific bilinear relation between the 3D structure parameters and the parameters describing the image motion. This relation enables the use of a fast factorization algorithm to solve simultaneously for the high number of unknown parameters describing the 3D structure. We refer to this approach as the *surface-based rank 1 factorization method*.

Maximum Likelihood estimate: continuation method

In chapter 10 we infer the 3D shape directly from the image intensity values, through ML estimation. We introduce the 3D motion estimates into the ML cost function and minimize it with respect to the unknown 3D shape.

To accomplish the minimization of the ML cost function, we use a computationally simple *multiresolution* approach – we develop a *continuation-type method* that works by refining the 3D shape estimate as more images are taken into account. Each step of the continuation method is solved by a *Gauss-Newton method* that requires no more than two or three iterations. The derivatives involved in the Gauss-Newton method are easily obtained in terms of the image gradients.

7.6 Summary

This chapter states the problem of recovering three-dimensional (3D) rigid structure from an image sequence. We review several approaches in the literature focusing on the references that are more closely related to our approach.

We describe the problem of inferring 3D rigid structure from an image sequence through an analogy with a classical communications system. We present the observation model and formulate the *Maximum Likelihood* (ML) estimate.

We minimize the ML cost function by first expressing the minimizer for the object texture in terms of the unknowns 3D shape and 3D motion. After replacing the texture estimate, we obtain a cost function that depends on the 3D structure through the 2D motion induced on the image plane.

The chapter ends with an outline of our approach to the problem of minimizing the ML cost function: first, we infer *structure from motion* by using the *surface-based rank 1 factorization method* described in chapters 8 and 9; then, we insert the 3D motion estimates into the ML cost function and minimize it with respect to the 3D shape by using the *multiresolution continuation-type method* described in chapter 10.

Chapter 8

Rank 1 Factorization

8.1 Introduction

In chapter 7 we showed that the *Maximum Likelihood* (ML) estimate of the three-dimensional (3D) structure from a two-dimensional (2D) video sequence leads to a cost function that depends on the 3D structure only through the 2D motion induced on the image plane. In this chapter we show how to recover the 3D rigid structure from the 2D motion by factorizing a matrix that is rank 1 in a noiseless situation.

The shape of the object is represented by the 3D positions of a set of feature points. As the object moves, the projection in the image plane of each feature point describes a different trajectory. The 2D motion induced on the image plane is then described by the set of trajectories of the feature point projections. We estimate each trajectory by computing the 2D translation of a small square around each feature point projection between successive frames. This step corresponds to the minimization of a simplified version of the ML cost function, as seen in chapter 3. We then recover the 3D shape and 3D motion of the object from the trajectories of the feature point projections by using an expedite method that exploits linear subspace constraints imposed by the rigidity of the scene.

Tomasi and Kanade pioneered the use of linear subspace constraints in motion analysis. They introduced the *factorization method*, see references [59, 60, 61], an elegant method to recover rigid structure from a set of trajectories of feature point projections.

In the factorization method the 3D shape and 3D motion are estimated by factorizing a measurement matrix whose entries are the set of trajectories of the feature point projections. The key idea underlying the factorization method is the fact that the rigidity of the scene imposes that the measurement matrix lives in a low dimensional subspace of the universe of matrices. Tomasi and Kanade have shown that the measurement matrix is a rank 3 matrix in a noiseless situation.

Exploiting further the linear subspace constraints, we solve the *structure from motion* (SFM) problem by factorizing a matrix that is rank 1 in a noiseless situation, rather than a rank 3 matrix as in the original factorization method. In our formulation, the unknowns are the 3D motion and the relative depths of the set of features, not their 3D positions as considered by Tomasi and Kanade in references [59, 61]. The coordinates of the features along the camera plane are given by their image positions in the first frame. The knowledge of the coordinates along the camera plane enables us to solve the SFM problem by factorizing a rank 1 matrix instead of a rank 3 matrix as in references [59, 61]. This simplifies the decomposition and normalization stages involved in the factorization approach.

Tomasi and Kanade [59, 61] use *Singular Value Decomposition* (SVD) to factorize the measurement matrix. We avoid the computation of the SVD by using a fast iterative method to compute the rank 1 matrix that best matches the data. Reference [44] introduced a recursive formulation for the original rank 3 factorization that also does not use SVD. The method in reference [44] stores and updates a matrix that becomes very large as the number of features increases. In our implementation, it is not necessary to store or compute any other matrix than the matrix to be factorized.

The chapter is organized as follows. In section 8.2 we formulate the SFM problem and review the original factorization method of Tomasi and Kanade [59, 60, 61]. Section 8.3 shows how to recover SFM by factorizing a rank 1 matrix. We detail the steps involved: decomposition and normalization. Section 8.4 deals with the situation for which the normalization procedure involved in the factorization approach fails. We derive the

analytical solution for this case and discuss its physical interpretation. The algorithm to compute the best rank 1 approximation is described in section 8.5. Section 8.6 describes one experiment that illustrates our approach and compares the computational cost of the rank 1 factorization with the cost of the original factorization method of Tomasi and Kanade [59, 60, 61]. Section 8.7 concludes the chapter.

8.2 Factorization Approach

We consider the same scenario of references [59, 61]. Figure 8.1 illustrates the scenario. The object coordinate system (o.c.s.) has axes labeled by x , y , and z . The shape of the object is described by the 3D position of a set of N feature points. The three-dimensional (3D) position of feature n is expressed in terms of the o.c.s. by (x_n, y_n, z_n) . The camera coordinate system (c.c.s.) has axes labeled by u , v , and w . The plane defined by the axes u and v is the camera plane.

The 3D motion of the object is defined by specifying the position of the o.c.s. $\{x, y, z\}$ relative to the c.c.s. $\{u, v, w\}$, i.e., by specifying a rotation-translation pair that takes values in the group of the rigid transformations of the space, the special Euclidean group $SE(3)$. The unconstrained 3D motion of a rigid body can be described in terms of a time varying point translation and a rotation. We express the object position at time instant f in terms of $(\mathbf{t}_f, \mathbf{\Theta}_f)$ where the vector $\mathbf{t}_f = [t_{uf}, t_{vf}, t_{wf}]^T$ contains the coordinates of the origin of the object coordinate system with respect to the camera coordinate system (translational component of the 3D motion), and $\mathbf{\Theta}_f$ is the rotation matrix that represents the orientation of the object coordinate system relative to the camera coordinate system (rotational component of the 3D motion). The matrix $\mathbf{\Theta}_f$ is determined by the Euler angles, the angles θ_f , ϕ_f , and ψ_f of three successive rotations, with respect to, respectively, the axes y , z , and x . The 3D rotation matrix $\mathbf{\Theta}$ is expressed in terms of the Euler angles θ ,

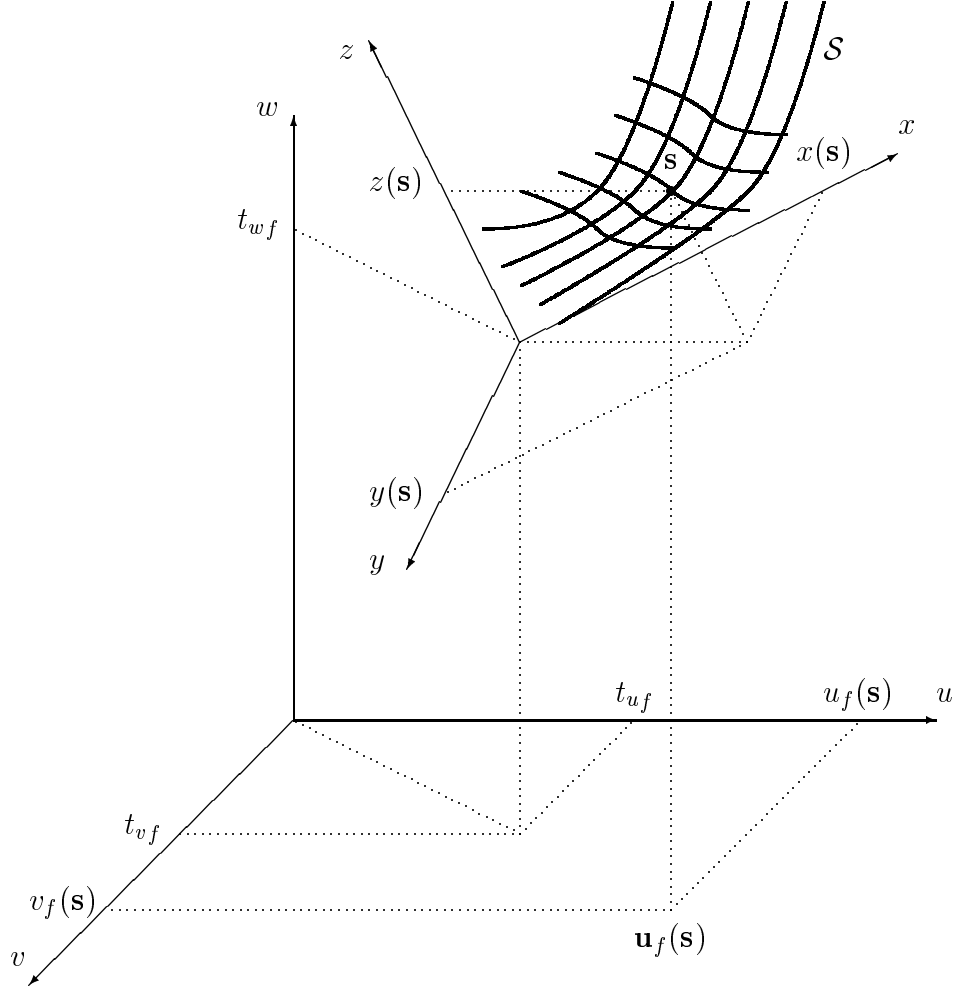


Figure 8.1: Factorization scenario: object and camera coordinate systems

ϕ , and ψ as, see reference [10],

$$\Theta = \begin{bmatrix} \cos \theta \cos \phi & \sin \phi & -\sin \theta \cos \phi \\ \sin \theta \sin \psi - \cos \theta \sin \phi \cos \psi & \cos \phi \cos \psi & \sin \theta \sin \phi \cos \psi + \cos \theta \sin \psi \\ \sin \theta \cos \psi + \cos \theta \sin \phi \sin \psi & -\cos \phi \sin \psi & \cos \theta \cos \psi - \sin \theta \sin \phi \sin \psi \end{bmatrix}. \quad (8.1)$$

At instant f , the point on the object with 3D coordinates (x, y, z) in the o.c.s. has the following coordinates in the c.c.s.,

$$\begin{bmatrix} u_f \\ v_f \\ w_f \end{bmatrix} = \Theta_f \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{t}_f = \begin{bmatrix} i_{xf} & i_{yf} & i_{zf} \\ j_{xf} & j_{yf} & j_{zf} \\ k_{xf} & k_{yf} & k_{zf} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_{uf} \\ t_{vf} \\ t_{wf} \end{bmatrix}, \quad (8.2)$$

where i_{xf} , i_{yf} , i_{zf} , j_{xf} , j_{yf} , j_{zf} , k_{xf} , k_{yf} , and k_{zf} are the entries of the rotation matrix Θ_f

in expression (8.1) and t_{uf} , t_{vf} , and t_{wf} are the entries of the translation vector \mathbf{t}_f .

We now make explicit the mapping $\mathbf{u}_f(\mathbf{s})$ that projects the point \mathbf{s} in the object surface to the point $\mathbf{u}_f(\mathbf{s})$ in the image plane, as introduced in the previous chapter and illustrated in Figure 7.1. We choose the texture coordinate vector \mathbf{s} to coincide with the coordinates of the plane defined by the axes x and y of the o.c.s., i.e., we choose $\mathbf{s} = [x, y]^T$. According to expression (8.2), the mapping $\mathbf{u}_f(\mathbf{s})$ is then

$$\mathbf{u}_f(\mathbf{s}) = \mathbf{u}_f \left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \begin{bmatrix} i_{xf} & i_{yf} & i_{zf} \\ j_{xf} & j_{yf} & j_{zf} \end{bmatrix} \begin{bmatrix} x \\ y \\ z(x, y) \end{bmatrix} + \begin{bmatrix} t_{uf} \\ t_{vf} \end{bmatrix}, \quad (8.3)$$

where $z(x, y)$ is the coordinate along the axis z of the o.c.s. of the point with coordinates $\mathbf{s} = [x, y]^T$ in the object surface. As mentioned in chapter 7, the projection mapping $\mathbf{u}_f(\mathbf{s})$ depends on the 3D motion of the object – the parameters i_{xf} , i_{yf} , i_{zf} , j_{xf} , j_{yf} , j_{zf} , t_{uf} , and t_{vf} in expression (8.3) – and the 3D shape of the object – the unknown relative depth $z(x, y)$ in expression (8.3). Expression (8.3) shows that the orthogonal projection is insensitive to the translation component t_{wf} of the object motion. This reflects the well known fact that, under orthography, the absolute depth (distance from the camera to the object) cannot be estimated.

In this chapter we describe the 3D shape of the object by the 3D position of a set of feature points. As the object moves, the projections of the features describe trajectories in the image plane, according to expression (8.3). We track a set of N feature points along the image sequence of F frames. To estimate the trajectories of the features, we compute the 2D translation of a small square \mathcal{R} around each feature point projection between successive frames, as detailed in chapter 3, section 3.4. We select the feature points by using a “trackability” criterion that imposes both a well conditioned 2D translation estimation and a low expected variance of the estimation error. The dependence of these characteristics on the image brightness pattern was studied in chapter 3, section 3.4. The feature selection criterion selects the N features that satisfy the following conditions: (i) the condition number $k(\mathbf{\Gamma}_{\mathcal{R}})$, given by expression (3.36), is below a threshold, typically set to 10; and, (ii) the N selected features have the lowest error variance σ_p^2 , given by

expression (3.43), by making the scale factor $\sigma_t^2 = 1$ among the feature candidates that satisfy (i).

The projection of feature n in frame f , denoted by (u_{fn}, v_{fn}) , is expressed in terms of the o.c.s. coordinates (x_n, y_n, z_n) and the 3D motion parameters as

$$\begin{cases} u_{fn} = i_{xf}x_n + i_{yf}y_n + i_{zf}z_n + t_{uf} \\ v_{fn} = j_{xf}x_n + j_{yf}y_n + j_{zf}z_n + t_{vf} \end{cases}, \quad (8.4)$$

where $i_{xf}, i_{yf}, i_{zf}, j_{xf}, j_{yf}, j_{zf}$ are entries of the 3D rotation matrix Θ_f in expression (8.1) and t_{uf} and t_{vf} are the components of the object translation along the camera plane. We choose the object coordinate system and the camera coordinate system so that they coincide in the first frame; so, we have

$$u_{1n} = x_n \quad \text{and} \quad v_{1n} = y_n. \quad (8.5)$$

The coordinates of the feature points on the camera plane $\{x_n, y_n, 1 \leq n \leq N\}$ are given by expression (8.5). We formulate the SFM problem as solving the overconstrained system of equations (8.4) with respect to the following set of unknowns: the 3D positions of the object for $2 \leq f \leq F$; and the relative depths $\{z_n, 1 \leq n \leq N\}$.

By choosing the origin of the object coordinate system to coincide with the centroid of the set of feature points ($\sum_n x_n = \sum_n y_n = \sum_n z_n = 0$), the *Least Squares* (LS) estimate of the translation is the centroid of the feature point projections,

$$\hat{t}_{uf} = \frac{1}{N} \sum_{n=1}^N u_{fn} \quad \text{and} \quad \hat{t}_{vf} = \frac{1}{N} \sum_{n=1}^N v_{fn}. \quad (8.6)$$

We now replace the translation estimates in the system of equations (8.4), and define the parameters

$$\tilde{u}_{fn} = u_{fn} - \frac{1}{N} \sum_{n=1}^N u_{fn} \quad \text{and} \quad \tilde{v}_{fn} = v_{fn} - \frac{1}{N} \sum_{n=1}^N v_{fn}. \quad (8.7)$$

We collect the parameters $\{\tilde{u}_{fn}, \tilde{v}_{fn}\}$, $\{i_{xf}, i_{yf}, i_{zf}, j_{xf}, j_{yf}, j_{zf}\}$, and $\{x_n, y_n, z_n\}$ in ma-

trices \mathbf{R} , \mathbf{M} , and \mathbf{S} as follows

$$\mathbf{R} = \begin{bmatrix} \tilde{u}_{21} & \tilde{u}_{22} & \cdots & \tilde{u}_{2N} \\ \tilde{v}_{21} & \tilde{v}_{22} & \cdots & \tilde{v}_{2N} \\ \tilde{u}_{31} & \tilde{u}_{32} & \cdots & \tilde{u}_{3N} \\ \tilde{v}_{31} & \tilde{v}_{32} & \cdots & \tilde{v}_{3N} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{u}_{F1} & \tilde{u}_{F2} & \cdots & \tilde{u}_{FN} \\ \tilde{v}_{F1} & \tilde{v}_{F2} & \cdots & \tilde{v}_{FN} \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} i_{x2} & i_{y2} & i_{z2} \\ j_{x2} & j_{y2} & j_{z2} \\ i_{x3} & i_{y3} & i_{z3} \\ j_{x3} & j_{y3} & j_{z3} \\ \vdots & \vdots & \vdots \\ i_{xF} & i_{yF} & i_{zF} \\ j_{xF} & j_{yF} & j_{zF} \end{bmatrix}, \quad (8.8)$$

$$\mathbf{S}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \\ z_1 & z_2 & \cdots & z_N \end{bmatrix}. \quad (8.9)$$

With the definitions above, we rewrite the equation system (8.4) in matrix format as

$$\mathbf{R} = \mathbf{M}\mathbf{S}^T. \quad (8.10)$$

Matrix \mathbf{R} is a $2(F-1) \times N$ rank deficient matrix. In a noiseless situation, \mathbf{R} is rank 3, reflecting the high redundancy in the data, due to the 3D rigidity of the object. The relation expressed in matrix format as in expression (8.10) was introduced by Tomasi and Kanade in the original formulation of the factorization, see references [59, 61]. In our formulation, the rows corresponding to frame 1 are not included in \mathbf{R} and \mathbf{M} , and we use the fact that the first two rows of \mathbf{S}^T are known from the feature projections in frame 1.

The factorization approach, see references [59, 61], finds a suboptimal solution to the problem of computing \mathbf{M} and \mathbf{S} from \mathbf{R} . This problem is formulated as the minimization

$$\min_{\mathbf{M}, \mathbf{S}} \|\mathbf{R} - \mathbf{M}\mathbf{S}^T\|_F, \quad (8.11)$$

where the solution space is constrained by the orthonormality of the rows of the matrix \mathbf{M} , see expressions (8.1) and (8.8). The operator $\|\cdot\|_F$ denotes the Frobenius norm [14]. In references [59, 61], the nonlinear minimization above was solved in two stages. The first stage, *decomposition stage*, solves $\mathbf{R} = \mathbf{M}\mathbf{S}^T$ in the least square sense by computing the *Singular Value Decomposition* (SVD) of the matrix \mathbf{R} and selecting the 3 largest singular values. From

$$\mathbf{R} \simeq \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \quad (8.12)$$

where \mathbf{U} is $2(F-1) \times 3$, $\mathbf{\Sigma}$ is diagonal 3×3 , and \mathbf{V}^T is $3 \times N$, the solution is

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{A} \quad (8.13)$$

$$\mathbf{S}^T = \mathbf{A}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^T \quad (8.14)$$

where \mathbf{A} is a non-singular 3×3 matrix. The second stage, *normalization stage*, computes \mathbf{A} by approximating the constraints imposed by the structure of the matrix \mathbf{M} .

Our formulation takes advantage of the fact that the first two rows $\{x_n, y_n\}$ of \mathbf{S}^T are known. These are known from the position of the features in the first frame. The problem is now reduced to, given \mathbf{R} , compute \mathbf{M} and $\{z_n\}$. By singling out the first view as the reference with respect to which the relative depth is defined, we can use the common approach to the inference of 3D structure from 2D video, i.e., we describe the unknown shape by the distances along the third dimension for each pixel of the image plane. This formulation seems to be in opposition to the idea behind the original factorization method as formulated by Tomasi and Kanade [59, 60, 61]. In their first paper, entitled “Shape and Motion without Depth” [60], the factorization method is motivated by emphasizing that when the object is far from the camera the depth can not be computed, and the 3D shape must be represented in terms of the set of coordinates $\{x_n, y_n, z_n\}$. In the following section we show that if the unknown shape is represented by the entities we really don’t know, i.e, by the *relative depths* $\{z_n\}$, the solution to the problem is greatly simplified.

8.3 Rank 1 Factorization

The problem of estimating the matrix \mathbf{M} and the vector $\{z_n\}$ from the matrix \mathbf{R} , although nonlinear, has a specific structure: it is a bilinear constrained *Least Squares* (LS) problem. The bilinear relation comes from expression (8.10), where the motion unknowns and the shape unknowns appear multiplied by each other, and the constraints are imposed by the orthonormality of the rows of the matrix \mathbf{M} , see expression (8.8). We perform in sequence the decomposition stage that solves the unconstrained bilinear problem, and the normalization stage.

Decomposition stage

Because the first two rows of the matrix \mathbf{S}^T are known, we show that the unconstrained bilinear problem $\mathbf{R} = \mathbf{M}\mathbf{S}^T$ is solved by the factorization of a rank 1 matrix, rather than a rank 3 matrix like in references [59, 61].

Define $\mathbf{M} = [\mathbf{M}_0, \mathbf{m}_3]$ and $\mathbf{S} = [\mathbf{S}_0, \mathbf{z}]$. The matrices \mathbf{M}_0 and \mathbf{S}_0 contain the first two columns of the matrices \mathbf{M} and \mathbf{S} , respectively, the vector \mathbf{m}_3 is the third column of \mathbf{M} , and the vector \mathbf{z} is the third column of \mathbf{S} . We decompose the relative depth vector \mathbf{z} into the component that belongs to the space spanned by the columns of \mathbf{S}_0 and the component orthogonal to this space as

$$\mathbf{z} = \mathbf{S}_0 \mathbf{b} + \mathbf{a}, \quad \text{with} \quad \mathbf{a}^T \mathbf{S}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}. \quad (8.15)$$

We rewrite the matrix \mathbf{R} by inserting expression (8.15) in expression (8.10), obtaining

$$\mathbf{R} = \mathbf{M}_0 \mathbf{S}_0^T + \mathbf{m}_3 \mathbf{b}^T \mathbf{S}_0^T + \mathbf{m}_3 \mathbf{a}^T. \quad (8.16)$$

The decomposition stage solves the matrix equation (8.16) with respect to the unknowns \mathbf{M}_0 , \mathbf{m}_3 , \mathbf{b} , and \mathbf{a} , ignoring the constraints imposed by the structure of the matrix \mathbf{M} . We formulate this problem as the unconstrained minimization

$$\min_{\mathbf{M}_0, \mathbf{m}_3, \mathbf{b}, \mathbf{a}} \left\| \mathbf{R} - \mathbf{M}_0 \mathbf{S}_0^T - \mathbf{m}_3 \mathbf{b}^T \mathbf{S}_0^T - \mathbf{m}_3 \mathbf{a}^T \right\|_F. \quad (8.17)$$

Since we know the matrix \mathbf{S}_0 , we eliminate the dependence of the expression (8.17) on \mathbf{M}_0 by solving the linear LS for \mathbf{M}_0 in terms of the other variables. We get

$$\widehat{\mathbf{M}}_0 = \mathbf{R} \mathbf{S}_0 (\mathbf{S}_0^T \mathbf{S}_0)^{-1} - \mathbf{m}_3 \mathbf{b}^T, \quad (8.18)$$

where we used the *Moore-Penrose pseudoinverse*, see reference [14], and the orthogonality between the vector \mathbf{a} and the columns of the matrix \mathbf{S}_0 , see expression (8.15). By replacing $\widehat{\mathbf{M}}_0$ given by expression (8.18) in expression (8.17), we get

$$\min_{\mathbf{m}_3, \mathbf{a}} \left\| \widetilde{\mathbf{R}} - \mathbf{m}_3 \mathbf{a}^T \right\|_F, \quad (8.19)$$

where

$$\tilde{\mathbf{R}} = \mathbf{R} \left[\mathbf{I} - \mathbf{S}_0 (\mathbf{S}_0^T \mathbf{S}_0)^{-1} \mathbf{S}_0^T \right]. \quad (8.20)$$

We see that the decomposition stage does not determine the vector \mathbf{b} . This is because the component of the vector \mathbf{z} that lives in the space spanned by the columns of \mathbf{S}_0 does not affect the space spanned by the columns of the entire matrix \mathbf{S} and the decomposition stage restricts only this latter space.

The solution for the vectors \mathbf{m}_3 and \mathbf{a} is given by the rank 1 matrix that best approximates $\tilde{\mathbf{R}}$. In a noiseless situation, $\tilde{\mathbf{R}}$ is rank 1, since we would get

$$\tilde{\mathbf{R}} = \mathbf{m}_3 \mathbf{a}^T \quad (8.21)$$

by replacing \mathbf{R} , given by expression (8.16), in expression (8.20). By computing the largest singular value of $\tilde{\mathbf{R}}$ and the associated singular vectors, we get

$$\begin{aligned} \tilde{\mathbf{R}} &\simeq \mathbf{u} \sigma \mathbf{v}^T \\ \hat{\mathbf{m}}_3 &= \alpha \mathbf{u} \\ \hat{\mathbf{a}}^T &= \frac{\sigma}{\alpha} \mathbf{v}^T \end{aligned} \quad (8.22)$$

where α is a normalizing scalar different from zero.

To compute \mathbf{u} , σ , and \mathbf{v} we could use *Singular Value Decomposition* (SVD), but the rank deficiency of $\tilde{\mathbf{R}}$ enables the use of less expensive algorithms, as detailed in section 8.4. This makes our decomposition stage simpler than the one in the original factorization method of Tomasi and Kanade [59, 61]. In fact, the matrix $\tilde{\mathbf{R}}$ in expression (8.20) is equal to the matrix \mathbf{R} multiplied by the orthogonal projector onto the orthogonal complement of the space spanned by the columns of \mathbf{S}_0 . This projection reduces the rank of the problem from 3 (matrix \mathbf{R}) to 1 (matrix $\tilde{\mathbf{R}}$).

Normalization stage

In this stage, we compute the scalar α and the vector \mathbf{b} by imposing the constraints that come from the structure of the matrix \mathbf{M} . The normalization stage is also simpler

than the one in Tomasi and Kanade [59, 61] because the number of unknowns is 3 (α and $\mathbf{b} = [b_1, b_2]^T$) as opposed to the 9 entries of a generic 3×3 normalization matrix.

By replacing the estimate $\widehat{\mathbf{m}}_3$, given by expression (8.22), in expression (8.18), we get for the estimate $\widehat{\mathbf{M}}$,

$$\widehat{\mathbf{M}} = \begin{bmatrix} \widehat{\mathbf{M}}_0 & \widehat{\mathbf{m}}_3 \end{bmatrix} = \mathbf{N} \begin{bmatrix} \mathbf{I}_{2 \times 2} & \mathbf{0}_{2 \times 1} \\ -\alpha \mathbf{b}^T & \alpha \end{bmatrix}, \quad (8.23)$$

where

$$\mathbf{N} = \begin{bmatrix} \mathbf{R} \mathbf{S}_0 (\mathbf{S}_0^T \mathbf{S}_0)^{-1} & \mathbf{u} \end{bmatrix}. \quad (8.24)$$

The constraints imposed by the structure of the matrix \mathbf{M} are the unit norm of each row, and the orthogonality between the consecutive rows $2j - 1$ and $2j$, see expressions (8.8) and (8.1). In terms of \mathbf{N} , α , and \mathbf{b} , the constraints are then

$$\mathbf{n}_i^T \begin{bmatrix} \mathbf{I}_{2 \times 2} & -\alpha \mathbf{b} \\ -\alpha \mathbf{b}^T & \alpha^2(1 + \mathbf{b}^T \mathbf{b}) \end{bmatrix} \mathbf{n}_i = 1, \quad 1 \leq i \leq 2(F - 1), \quad (8.25)$$

$$\mathbf{n}_{2j-1}^T \begin{bmatrix} \mathbf{I}_{2 \times 2} & -\alpha \mathbf{b} \\ -\alpha \mathbf{b}^T & \alpha^2(1 + \mathbf{b}^T \mathbf{b}) \end{bmatrix} \mathbf{n}_{2j} = 0, \quad 1 \leq j \leq F - 1, \quad (8.26)$$

where \mathbf{n}_i^T denotes the row i of the matrix \mathbf{N} .

We compute the normalization parameters α and \mathbf{b} in an analogous way to the one of the original factorization method of Tomasi and Kanade [59, 61]. To compute their normalization matrix \mathbf{A} , they solve first a linear LS problem that determines the intermediate matrix $\mathbf{B} = \mathbf{A} \mathbf{A}^T$. Then the normalization matrix \mathbf{A} is computed from the estimate $\widehat{\mathbf{B}}$ of the matrix \mathbf{B} . We also compute the normalization parameters α and \mathbf{b} from the linear LS solution of the system of equations (8.25, 8.26).

To compute the linear LS solution of the system of equations (8.25, 8.26), we define the intermediate parameters ϵ_1 , ϵ_2 , and ϵ_3 as the unknown entries of the matrix that appears in expressions (8.25) and (8.26),

$$\epsilon_1 = -\alpha b_1, \quad \epsilon_2 = -\alpha b_2, \quad \epsilon_3 = \alpha^2(1 + b_1^2 + b_2^2). \quad (8.27)$$

The left hand side of equations (8.25) and (8.26) depends linearly on the parameters ϵ_1 ,

ϵ_2 , and ϵ_3 . Carrying out the matrix product, we get

$$\mathbf{n}_i^T \left[\begin{array}{c|c} \mathbf{I}_{2 \times 2} & \begin{matrix} \epsilon_1 \\ \epsilon_2 \end{matrix} \\ \hline \begin{matrix} \epsilon_1 & \epsilon_2 \end{matrix} & \epsilon_3 \end{array} \right] \mathbf{n}_j = (n_{i1}n_{j3} + n_{i3}n_{j1})\epsilon_1 + (n_{i2}n_{j3} + n_{i3}n_{j2})\epsilon_2 + n_{i3}n_{j3}\epsilon_3 + n_{i1}n_{j1} + n_{i2}n_{j2}, \quad (8.28)$$

where n_{ij} denotes the element (i, j) of matrix \mathbf{N} .

The system of equations (8.25, 8.26) is written in terms of the parameters ϵ_1 , ϵ_2 , and ϵ_3 as the over-constrained linear system

$$\Xi \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \boldsymbol{\xi} \quad (8.29)$$

where the $3(F-1) \times 3$ matrix Ξ and the $3(F-1) \times 1$ vector $\boldsymbol{\xi}$ collect functions of the entries of the matrix \mathbf{N} . According to the expression (8.28), we get

$$\Xi = \left[\begin{array}{ccc} 2n_{11}n_{13} & 2n_{12}n_{13} & n_{13}^2 \\ 2n_{21}n_{23} & 2n_{22}n_{23} & n_{23}^2 \\ \vdots & \vdots & \vdots \\ 2n_{2F-2,1}n_{2F-2,3} & 2n_{2F-2,2}n_{2F-2,3} & n_{2F-2,3}^2 \\ \hline n_{11}n_{23} + n_{13}n_{21} & n_{12}n_{23} + n_{13}n_{22} & n_{13}n_{23} \\ n_{31}n_{43} + n_{33}n_{41} & n_{32}n_{43} + n_{33}n_{42} & n_{33}n_{43} \\ \vdots & \vdots & \vdots \\ n_{2F-3,1}n_{2F-2,3} + n_{2F-3,3}n_{2F-2,1} & n_{2F-3,2}n_{2F-2,3} + n_{2F-3,3}n_{2F-2,2} & n_{2F-3,3}n_{2F-2,3} \end{array} \right], \quad (8.30)$$

$$\boldsymbol{\xi} = \left[\begin{array}{c} 1 - n_{11}^2 - n_{12}^2 \\ 1 - n_{21}^2 - n_{22}^2 \\ \vdots \\ 1 - n_{2F-2,1}^2 - n_{2F-2,2}^2 \\ \hline n_{11}n_{21} - n_{12}n_{22} \\ n_{31}n_{41} - n_{32}n_{42} \\ \vdots \\ n_{2F-3,1}n_{2F-2,1} - n_{2F-3,2}n_{2F-2,2} \end{array} \right]. \quad (8.31)$$

We compute the parameters ϵ_1 , ϵ_2 , and ϵ_3 as the LS solution of the over-constrained linear system (8.29).

To complete the normalization stage, we compute the estimates of the scalar α and the vector $\mathbf{b} = [b_1, b_2]^T$ from the parameters ϵ_1 , ϵ_2 , and ϵ_3 by inverting the relations in expression (8.27). We obtain

$$|\hat{\alpha}| = \sqrt{\epsilon_3 - \epsilon_1^2 - \epsilon_2^2}, \quad \hat{b}_1 = \epsilon_1/\hat{\alpha}, \quad \hat{b}_2 = \epsilon_2/\hat{\alpha}. \quad (8.32)$$

There is an ambiguity in the sign of α , since both the positive and negative solutions of the square root are valid solutions in expression (8.32). This ambiguity is inherent to the orthographic projection model. In fact, the trajectories of the feature points are equally explained by two different 3D rigid structures that are as follows. Consider an object with relative depth $\bar{\mathbf{z}} = -\mathbf{z}$ (mirror reflection) and whose motion is such that the third column of the rotation matrix is $\bar{\mathbf{m}}_3 = -\mathbf{m}_3$. In this scenario, according to (8.15), we have $\bar{\mathbf{b}} = -\mathbf{b}$ and $\bar{\mathbf{a}} = -\mathbf{a}$. It is clear that this scenario corresponds to a change on the sign of α . Expressions (8.22) and (8.32) explain why the change on the sign of α implies the change on the signs of \mathbf{m}_3 , \mathbf{a} , and \mathbf{b} . Now see from equation system (8.4) that the trajectories of the feature points for this second scenario are the same as for the original scenario. This is because all the entities on the right hand side of the equations (8.4) are the same for the two scenarios, except for $\bar{i}_{zf} = -i_{zf}$, $\bar{j}_{zf} = -j_{zf}$, and $\bar{z}_n = -z_n$, whose product cancels the minus signs. The reader may wonder if it is possible, given \mathbf{M} , to define a different 3D motion such that the third column of the rotation matrix is $\bar{\mathbf{m}}_3 = -\mathbf{m}_3$ and the first two columns of the rotation matrix are $\bar{\mathbf{m}}_1 = \mathbf{m}_1$ and $\bar{\mathbf{m}}_2 = \mathbf{m}_2$. In fact, it is always possible to find such a motion by defining the Euler angles $\bar{\theta}_f = -\theta_f$, $\bar{\phi}_f = \phi_f$, and $\bar{\psi}_f = -\psi_f$; see from expressions (8.1) and (8.8) that the first two columns of \mathbf{M} remain the same and the sign of the third column becomes the opposite.

8.4 Normalization Failure

If the values of the intermediate parameters ϵ_1 , ϵ_2 , and ϵ_3 that come from the *Least Squares* (LS) solution of the system (8.29) are such that

$$\epsilon_3 < \epsilon_1^2 + \epsilon_2^2, \quad (8.33)$$

the scalar α and the vector $\mathbf{b} = [b_1, b_2]^T$ can not be recovered by expression (8.32). This situation corresponds to what is called a *normalization failure* in the original factorization method of Tomasi and Kanade [59, 61]. In references [59, 61], the normalization stage computes the normalization matrix \mathbf{A} by solving first a LS problem that determines the intermediate matrix $\mathbf{B} = \mathbf{A}\mathbf{A}^T$. Then the normalization matrix \mathbf{A} is computed from the estimate $\hat{\mathbf{B}}$ of the matrix \mathbf{B} . The normalization failure situation is when there is no matrix $\hat{\mathbf{A}}$ such that $\hat{\mathbf{A}}\hat{\mathbf{A}}^T = \hat{\mathbf{B}}$, i.e., matrix $\hat{\mathbf{B}}$ is not nonnegative definite.

The normalization procedure described in section 8.3 is an expedite way to compute the normalization parameters α and $\mathbf{b} = [b_1, b_2]^T$ that best match the restrictions imposed by the structure of \mathbf{M} . However, in the special cases where the linear LS solution for the intermediate parameters ϵ_1 , ϵ_2 , and ϵ_3 do not correspond to valid values for α and \mathbf{b} that procedure does not provide estimates for the normalization parameters.

In this section, we study the estimation of the normalization parameters α and \mathbf{b} directly from the constraints imposed by the structure of the matrix \mathbf{M} , rather than using any intermediate parameters. We will see that whenever the normalization procedure described above fails, the direct estimation of the normalization parameters α and \mathbf{b} leads to the estimate $\hat{\alpha} = 0$.

To derive the LS estimate of the normalization parameters α and $\mathbf{b} = [b_1, b_2]^T$ directly from the constraints imposed by the structure of the matrix \mathbf{M} , i.e., directly from the system of equations (8.25,8.26), we make explicit the LS cost function involved in the linear system of expression (8.29),

$$C_N = \left(\Xi \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} - \boldsymbol{\xi} \right)^T \left(\Xi \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} - \boldsymbol{\xi} \right). \quad (8.34)$$

The direct minimization of the cost function C_N in expression (8.34) with respect to the normalization parameters α and $\mathbf{b} = [b_1, b_2]^T$ requires the partial derivatives of C_N with respect to α , b_1 , and b_2 to be zero. These derivatives are expressed in terms of the

partial derivatives of C_N with respect to the intermediate parameters ϵ_1 , ϵ_2 , and ϵ_3 as

$$\frac{\partial C_N}{\partial b_1} = \frac{\partial C_N}{\partial \epsilon_1} \frac{\partial \epsilon_1}{\partial b_1} + \frac{\partial C_N}{\partial \epsilon_3} \frac{\partial \epsilon_3}{\partial b_1} = -\frac{\partial C_N}{\partial \epsilon_1} \alpha + 2 \frac{\partial C_N}{\partial \epsilon_3} b_1 \alpha^2, \quad (8.35)$$

$$\frac{\partial C_N}{\partial b_2} = \frac{\partial C_N}{\partial \epsilon_2} \frac{\partial \epsilon_2}{\partial b_2} + \frac{\partial C_N}{\partial \epsilon_3} \frac{\partial \epsilon_3}{\partial b_2} = -\frac{\partial C_N}{\partial \epsilon_2} \alpha + 2 \frac{\partial C_N}{\partial \epsilon_3} b_2 \alpha^2, \quad (8.36)$$

$$\frac{\partial C_N}{\partial \alpha} = \frac{\partial C_N}{\partial \epsilon_3} \frac{\partial \epsilon_3}{\partial \alpha} = 2 \frac{\partial C_N}{\partial \epsilon_3} \alpha (1 + b_1^2 + b_2^2). \quad (8.37)$$

The normalization procedure of section 8.3 solves

$$\frac{\partial C_N}{\partial b_1} = \frac{\partial C_N}{\partial b_2} = \frac{\partial C_N}{\partial \alpha} = 0 \quad (8.38)$$

by equating to zero the partial derivatives

$$\frac{\partial C_N}{\partial \epsilon_1} = \frac{\partial C_N}{\partial \epsilon_2} = \frac{\partial C_N}{\partial \epsilon_3} = 0 \quad (8.39)$$

as it comes from the LS solution of the linear system (8.29). If the values of the parameters ϵ_1 , ϵ_2 , and ϵ_3 that satisfy (8.39) correspond, through the relations (8.27), to valid values of the normalization parameters α , b_1 , and b_2 , these values of α , b_1 , and b_2 are the ones that minimize the cost function C_N in expression (8.34). If the values of the parameters ϵ_1 , ϵ_2 , and ϵ_3 that satisfy (8.39) do not correspond to valid values of the normalization parameters, the minimizers $\hat{\alpha}$, \hat{b}_1 , and \hat{b}_2 of the cost function C_N are such that at least one of the partial derivatives $\frac{\partial C_N}{\partial \epsilon_1}$, $\frac{\partial C_N}{\partial \epsilon_2}$, and $\frac{\partial C_N}{\partial \epsilon_3}$ is different from zero and the partial derivatives $\frac{\partial C_N}{\partial b_1}$, $\frac{\partial C_N}{\partial b_2}$, and $\frac{\partial C_N}{\partial \alpha}$ are all zero.

If the partial derivative $\frac{\partial C_N}{\partial \epsilon_3}$ is different from zero, the estimate $\hat{\alpha}$ must be zero so that the partial derivative $\frac{\partial C_N}{\partial \alpha}$ in expression (8.37) is zero. The quantities \hat{b}_1 and \hat{b}_2 can take any value because $\frac{\partial C_N}{\partial b_1} = 0$ and $\frac{\partial C_N}{\partial b_2} = 0$ from expressions (8.35) and (8.36). If $\frac{\partial C_N}{\partial \epsilon_3} = 0$ and $\frac{\partial C_N}{\partial \epsilon_1} \neq 0$, $\hat{\alpha}$ must be zero to make the partial derivative $\frac{\partial C_N}{\partial b_1}$ in expression (8.35) to equal zero and again \hat{b}_2 and \hat{b}_1 can take any value because $\frac{\partial C_N}{\partial b_2} = 0$ and $\frac{\partial C_N}{\partial \alpha} = 0$ from expressions (8.36) and (8.37). If $\frac{\partial C_N}{\partial \epsilon_3} = 0$ and $\frac{\partial C_N}{\partial \epsilon_2} \neq 0$, we conclude in an analogous way that $\hat{\alpha}$ must be zero and \hat{b}_1 and \hat{b}_2 can take any value. In this way, we conclude that, whenever the normalization procedure of section 8.3 fails, the direct estimation of the

normalization parameters α and $\mathbf{b} = [b_1, b_2]^T$ leads to $\hat{\alpha} = 0$ and does not restrict the vector $\hat{\mathbf{b}}$.

When the estimate of the normalization parameter α is $\hat{\alpha} = 0$, this is an indication that the method of performing the sequence of stages decomposition-normalization does not work. This is because the parameter α was introduced in the factorization (8.22) as a normalizing scalar different from zero. In fact, when the estimate $\hat{\alpha}$ is zero, it is an indication that the matrix $\tilde{\mathbf{R}}$ in expressions (8.20,8.21,8.22) is not well approximated by a rank 1 matrix. This situation may arise due to two different reasons. First, the scene may contain dramatic perspective effects that lead to a matrix $\tilde{\mathbf{R}}$ of rank higher than 1 even in a noiseless situation. Second, the 3D shape of the object or its 3D motion can be such that the matrix $\tilde{\mathbf{R}}$ is $\mathbf{0}$ in a noiseless situation. The first situation can be solved only by taking into account in the analysis the perspective projection. The second situation occurs with the following degenerate cases: when the 3D motion is such that the third column of the matrix \mathbf{M} is $\mathbf{m}_3 = \mathbf{0}$; or when the 3D shape is planar, i.e., the 3D shape is such that

$$\forall_n : z_n = \gamma_x x_n + \gamma_y y_n. \quad (8.40)$$

This is equivalent to say that the vector \mathbf{a} in expression (8.15) is $\mathbf{a} = \mathbf{0}$ and the relative depth vector \mathbf{z} is $\mathbf{z} = \mathbf{S}_0 \mathbf{b}$. For any of these degenerate cases, we get

$$\mathbf{R} = \mathbf{M}_0 \mathbf{S}_0^T + \mathbf{m}_3 \mathbf{b}^T \mathbf{S}_0^T \quad (8.41)$$

in expression (8.16), and $\tilde{\mathbf{R}} = \mathbf{0}$ in expressions (8.20,8.21). In these degenerate cases there is not enough information in the feature trajectories to recover the 3D structure. In spite of this, the images in the sequence can still be aligned by computing $\widehat{\mathbf{M}}_0$ according to expression (8.18) for any choice of \mathbf{m}_3 and \mathbf{b} , for example, by making

$$\widehat{\mathbf{M}}_0 = \mathbf{R} \mathbf{S}_0 (\mathbf{S}_0^T \mathbf{S}_0)^{-1}. \quad (8.42)$$

In reference [47], the author uses a numerical technique to compute the normalization matrix \mathbf{A} involved in the original rank 3 factorization method, when the normalization

procedure of Tomasi and Kanade [59, 61] fails. The numerical technique in reference [47] estimates the normalization matrix \mathbf{A} directly from the constraints imposed by the structure of the matrix \mathbf{M} , rather than using the intermediate matrix $\mathbf{B} = \mathbf{A}\mathbf{A}^T$. The author concludes by experimentation that whenever the normalization procedure fails, the numerical method used to estimate directly the normalization matrix converges to a singular matrix \mathbf{A} . In reference [47] this is considered to be a degenerate case where the measurement matrix \mathbf{R} can be approximated by a matrix of rank less than 3. In our case, we were able to derive analytically that whenever the normalization procedure fails, the direct estimation of the normalization parameters α and $\mathbf{b} = [b_1, b_2]^T$ leads to $\hat{\alpha} = 0$. The simple analytical derivation above explains the experimental evidence reported in reference [47].

8.5 Rank 1 Approximation: Iterative Algorithm

This section describes how we compute the rank 1 approximation of a given matrix $\tilde{\mathbf{R}}$. The problem is written as

$$\min_{\mathbf{u}, \mathbf{v}} \left\| \tilde{\mathbf{R}} - \mathbf{u}\mathbf{v}^T \right\|_F, \quad (8.43)$$

which comes from expression (8.22) by including the scaling factor σ into the vector \mathbf{v} . The solution to the problem (8.43) is known to be given by the *Singular Value Decomposition* (SVD) of the matrix $\tilde{\mathbf{R}}$ after selecting the largest singular value, see reference [27].

The rank deficiency of the matrix $\tilde{\mathbf{R}}$ enables the use of a less expensive iterative technique to compute the decomposition. It is based on the fact that only the right singular vector \mathbf{v} that corresponds to the largest singular value has to be computed. Since the vector \mathbf{v} is the eigenvector of the matrix $\tilde{\mathbf{R}}^T \tilde{\mathbf{R}}$ that corresponds to the largest eigenvalue, we start with a random choice \mathbf{v}_0 and iterate,

$$\mathbf{v}_{i+1} = \frac{\mathbf{v}_i^T \tilde{\mathbf{R}}}{\mathbf{v}_i^T \tilde{\mathbf{R}}^T \tilde{\mathbf{R}} \mathbf{v}_i} \tilde{\mathbf{R}}^T \tilde{\mathbf{R}} \mathbf{v}_i \quad (8.44)$$

until convergence, see reference [27]. In each iteration, the component of \mathbf{v}_i along the vector \mathbf{v} is more magnified than the components along the other eigenvectors of $\tilde{\mathbf{R}}^T \tilde{\mathbf{R}}$. The fraction in expression (8.44) is a normalizing factor. The left singular vector \mathbf{u} is

then computed by solving (8.43) with \mathbf{v} given by the final value of the iterative process above,

$$\mathbf{u} = \frac{\tilde{\mathbf{R}}\mathbf{v}}{\mathbf{v}^T\mathbf{v}}. \quad (8.45)$$

The speed of convergence of the iterative processes in expression (8.44) depends on the ratio of the largest eigenvector of the matrix $\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}$ by the second largest one. The higher that ratio, the faster is the convergence. When $\tilde{\mathbf{R}}$ is well approximated by a rank 1 matrix, as it is in our case, the iterative process in expression (8.44) converges in a few iterations. We stopped the iterations when the ∞ -norm of the difference between the vectors \mathbf{v}_i and \mathbf{v}_{i+1} is below a threshold.

This iterative procedure can be generalized to compute the best rank r approximation of a given matrix, for $r > 1$. This was done in reference [44] where the authors propose a recursive formulation to the original rank 3 factorization of Tomasi and Kanade [59, 61]. Their method stores and updates the $N \times N$ matrix $\mathbf{R}^T\mathbf{R}$. Then, they use this matrix to compute, iteratively, the best rank 3 approximation of the matrix \mathbf{R} . When the number N of features is large, the matrix $\mathbf{R}^T\mathbf{R}$ becomes very large. In our implementation of the decomposition stage, instead of computing the matrix $\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}$, we split the computation of each iteration in expression (8.44) by first computing $\tilde{\mathbf{R}}\mathbf{v}_i$,

$$\tilde{\mathbf{v}}_i = \tilde{\mathbf{R}}\mathbf{v}_i, \quad \mathbf{v}_{i+1} = \frac{\mathbf{v}_i^T\mathbf{v}_i}{\tilde{\mathbf{v}}_i^T\tilde{\mathbf{v}}_i}\tilde{\mathbf{R}}^T\tilde{\mathbf{v}}_i. \quad (8.46)$$

This way, we avoid the need to store or compute any other matrix than the matrix to be factorized, $\tilde{\mathbf{R}}$.

This approach easily extends to the rank 3 factorization of a matrix \mathbf{R} . In this case, the matrix \mathbf{V} that collects the three right singular vectors that correspond to the three largest singular values of \mathbf{R} is computed by iterating

$$\tilde{\mathbf{V}}_i = \mathbf{R}\mathbf{V}_i, \quad \mathbf{V}_{i+1} = \mathbf{R}^T\tilde{\mathbf{V}}_i \left(\tilde{\mathbf{V}}_i^T\tilde{\mathbf{V}}_i \right)^{-1} \left(\mathbf{V}_i^T\mathbf{V}_i \right) \quad (8.47)$$

until convergence. Then the matrix \mathbf{U} that collects the three left singular vectors is given by

$$\mathbf{U} = \mathbf{R}\mathbf{V} \left(\mathbf{V}^T\mathbf{V} \right)^{-1} \quad (8.48)$$

where \mathbf{V} is the final value of the iterative process (8.47).

In the comparative tests described in the following section, we use this iterative method to perform the decomposition step within the original rank 3 factorization method. We show that the computational cost of the iterative approach is much lower than the cost of performing SVD as suggested originally by Tomasi and Kanade [59, 61].

8.6 Experiments

We describe one experiment that uses synthetic data to illustrate the properties of the rank 1 matrix $\tilde{\mathbf{R}}$. Then we compare the computational cost of our approach with the one of the original factorization method. Experiments with real life video sequences are described in chapter 11.

Rank 1 factorization

We generated a set of 10 feature points randomly located inside a cube. The three-dimensional (3D) rotational motion was simulated by synthesizing a smooth time evolution for the Euler angles that specify the orientation of the object coordinate system relative to the camera coordinate system. We used the perspective projection model to project the features onto the image plane. The distance of the camera to the centroid of the set of feature points was set to a value high enough (approximately 10 times the maximum relative depth) such that orthographic projection can be considered to be a valid approximation. Figure 8.2 shows the feature trajectories on the image plane obtained for a set of 50 frames, after adding noise. For each trajectory, the initial position is marked with “o” and the final position is marked with “*”. The trajectories in Figure 8.2 are a representation of the columns of the matrix \mathbf{R} . The trajectory for feature n shows the n^{th} column of \mathbf{R} , i.e., it is the evolution of the image point $(\mathbf{R}_{2f-1,n}, \mathbf{R}_{2f,n})$ across the frame index f , see expression (8.8). The challenge in recovering *Structure from Motion* (SFM) comes from the fact that the 3D shape and the 3D motion are observed in a coupled way through the 2D motion on the image plane (the trajectories in Figure 8.2).

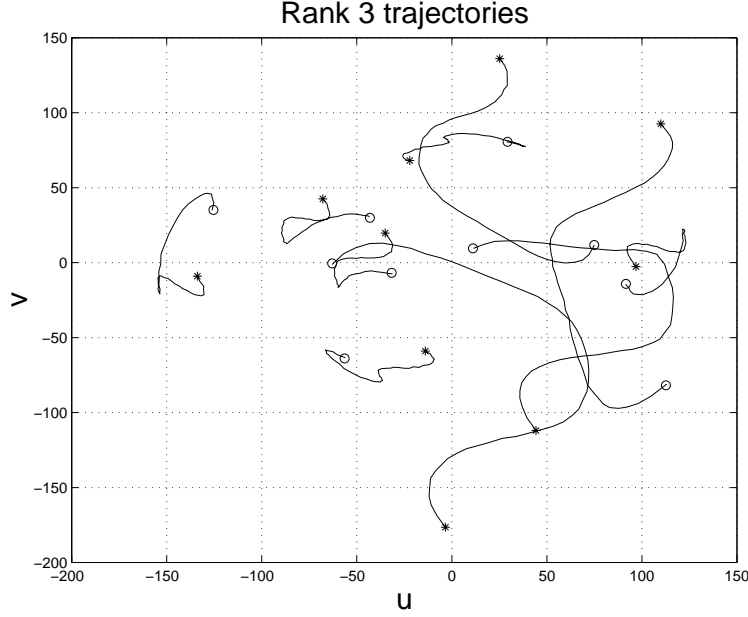


Figure 8.2: Feature trajectories on image plane.

From the data in matrix \mathbf{R} , we computed matrix $\tilde{\mathbf{R}}$ given by expression (8.20). The left side plot of Figure 8.3 represents the columns of $\tilde{\mathbf{R}}$ in the same way as Figure 8.2 plots \mathbf{R} , i.e., it shows the evolution of $(\tilde{\mathbf{R}}_{2f-1,n}, \tilde{\mathbf{R}}_{2f,n})$ across the frame index f , for each feature n . We see that all trajectories on the left side plot of Figure 8.3 have equal shape, unlike the ones in Figure 8.2. This is because we have eliminated the dependence of the trajectories on the x and y coordinates of the features, by making the subspace projection of expression (8.20). Each trajectory on the left side plot of Figure 8.3 is a scaled version of a fixed trajectory that does not depend on the object shape. This fixed trajectory is determined uniquely by the 3D motion of the object; it corresponds to the third column of the matrix \mathbf{M} , the vector \mathbf{m}_3 , see expression (8.21). The vector \mathbf{m}_3 is represented on the right side plot of Figure 8.3 in the same way as the columns of the matrices \mathbf{R} and $\tilde{\mathbf{R}}$. Compare the left side plot with the right side plot of Figure 8.3 to confirm the similarity of the shape of the trajectories. The scaling factor for each trajectory in matrix $\tilde{\mathbf{R}}$ depends on the relative depth z of the corresponding feature point, see expressions (8.21) and (8.15). Note from the plots in Figure 8.3 that some trajectories in the matrix $\tilde{\mathbf{R}}$ are symmetric

to the trajectory of the vector \mathbf{m}_3 , with respect to the origin of the image coordinates. These trajectories correspond to features for which the scaling factor that comes from their relative depth is negative.

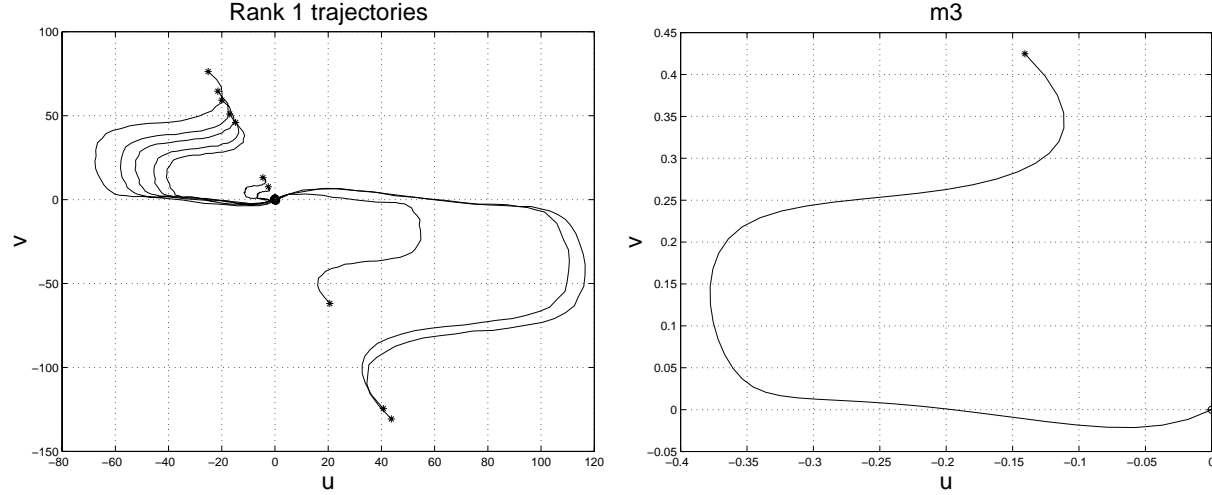


Figure 8.3: Left plot: trajectories from the columns of matrix $\tilde{\mathbf{R}}$. Right plot: trajectory from the third column of matrix \mathbf{M} .

This experiment illustrates that the subspace projection of expression (8.20) decouples the influence on the feature trajectories of the 3D motion from the influence of the 3D shape. In contrast to the trajectories of \mathbf{R} in Figure 8.2, for $\tilde{\mathbf{R}}$ in Figure 8.3, the 3D motion influences only the 2D shape of the trajectories, and the 3D shape influences only the magnitude of the trajectories.

Figure 8.4 plots the 10 larger singular values of the matrices \mathbf{R} , marked with “o”, and $\tilde{\mathbf{R}}$, marked with “*”. While the 3 larger singular values of the matrix \mathbf{R} contain the most of the energy, the matrix $\tilde{\mathbf{R}}$ is well described by only its largest singular value. The plots in Figure 8.4 confirm the analysis of sections 8.2 and 8.3 and the comment in the previous paragraphs.

Computational cost

To compare the computational cost of the rank 1 factorization algorithm with the one of the original factorization method, we ran the experiment described for a fixed number of

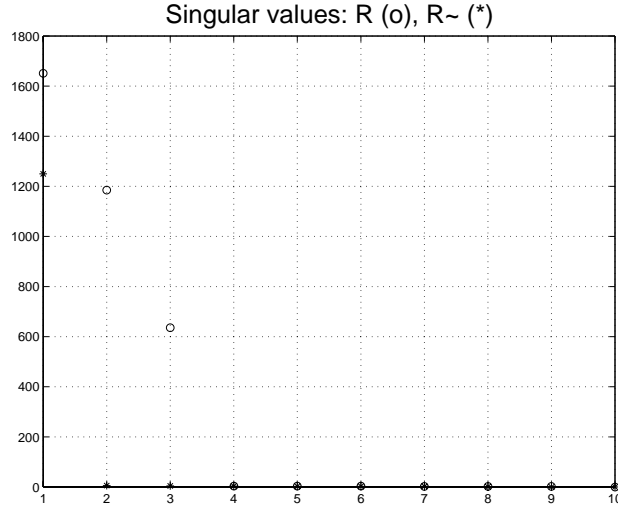


Figure 8.4: Singular values of matrices \mathbf{R} and $\tilde{\mathbf{R}}$.

$F = 50$ frames and a number of N feature points varying from 10 to 100; and for a fixed number of $N = 50$ feature points and a number of F frames varying from 10 to 100. We computed the average number of MatLab[®]¹ floating point operations (FLOPS) over 1000 tests for each experiment.

For each experiment, we estimate the 3D shape and 3D motion by using three methods: i) the original factorization method of Tomasi and Kanade [59, 61] that computes the *Singular Value Decomposition* (SVD) of the measurement matrix \mathbf{R} ; ii) the same method but computing the factorization of the rank 3 matrix \mathbf{R} by using the iterative procedure described in section 8.5, see expression (8.47); and iii) our formulation of the factorization as a rank 1 problem. The reason why we include method ii) in the experiment is because it is the fastest way available to compute the rank 3 factorization, making fair the comparison with the method iii). It is worth mentioning that the method ii) is not the recursive formulation presented in reference [44]. The method in reference [44] is well suited for a fixed small number of feature points and an increasing number of frames. On the other hand, with a large number of feature points, the recursive method in [44] becomes useless because it uses a matrix that becomes prohibitively large. As discussed in section 8.5, the

¹MatLab is a registered trademark of MathWorks.

storage space and computational power required by the method of reference [44] are then much higher than the ones required by methods ii) and iii) above.

Figures 8.5 and 8.6 plot the average number of FLOPS as a function of the number of frames and the number of feature points, for each of the three methods. The number of FLOPS are marked with dotted lines for method i), dashdotted lines for method ii), and solid lines for method iii). The left plots show the three curves, while the right plots show only the curves for methods ii) and iii) using a different vertical scale, for better visualization.

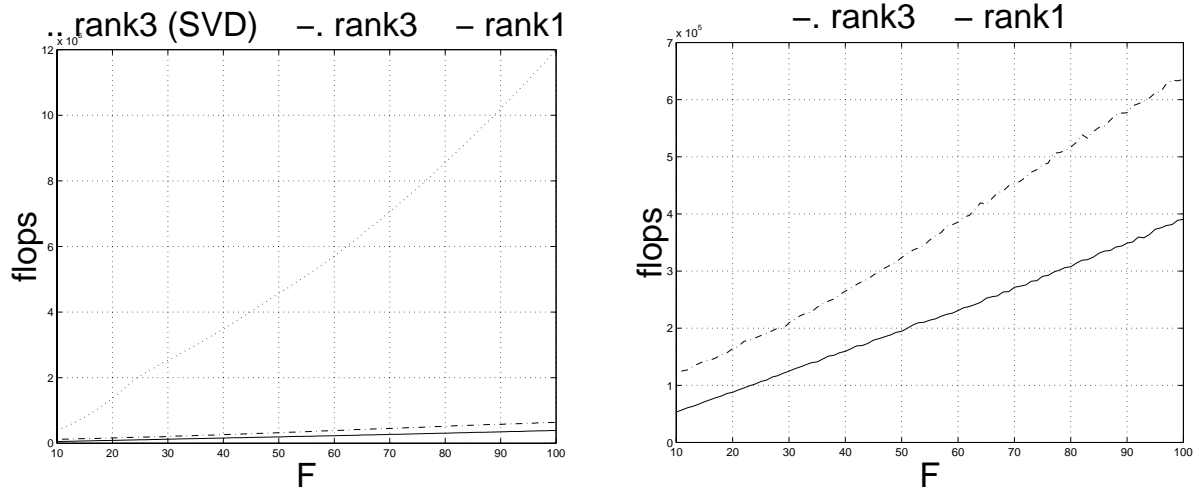


Figure 8.5: MatLab FLOPS count as a function of the number of frames.

From the left side plots, we see that the number of FLOPS is much larger for the original factorization method than for the iterative approaches. This is due to the high computational cost of the SVD. From the right side plots, we see that the number of FLOPS increases approximately linearly with both the number of frames and the number of feature points, for both iterative methods ii) and iii). The increasing rate is lower for the factorization of the rank 1 matrix $\tilde{\mathbf{R}}$ than the rank 3 matrix \mathbf{R} , by a factor of approximately 2. This is because both the decomposition and normalization stages in method iii) are simpler than the ones in method ii). In all the experiments, the performance of the three methods in terms of the accuracy of the estimates of the 3D structure is the same.

When the video sequence contains perspective distortions, the orthographic projection

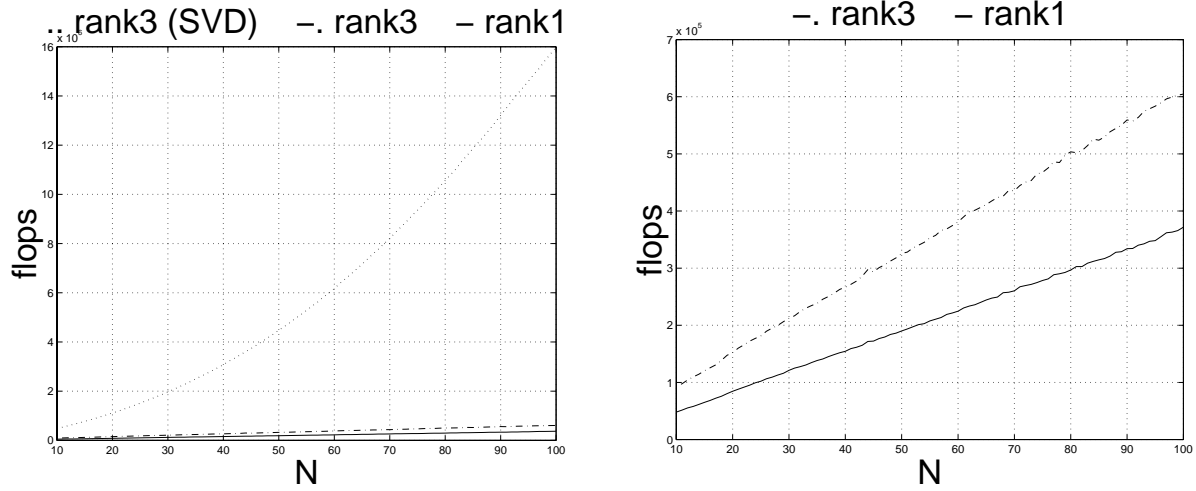


Figure 8.6: MatLab FLOPS count as a function of the number of feature points.

model is not a good approximation to the real perspective projection. In this case, the rank of the measurement matrix \mathbf{R} is larger than 3, even in a noiseless situation, due to the different structure of the feature trajectories, and the rank of the matrix $\tilde{\mathbf{R}}$ is larger than 1. In this case, we may expect that the observed trajectories are better approximated by the rank 3 factorization of \mathbf{R} than by the rank 1 factorization of $\tilde{\mathbf{R}}$ because there is no reason to believe that the first two rows of \mathbf{S}^T are singular vectors of \mathbf{R} , i.e., there is no reason to believe that the subspace projection (8.20) is the best way to reduce the rank of the matrix. In spite of this, the experiments we conducted lead to similar performances in terms of the accuracy of the estimates of the 3D structure for both the rank 1 factorization of $\tilde{\mathbf{R}}$ and the rank 3 factorization of \mathbf{R} . This behavior is because the better approximation capability of the rank 3 factorization is not based on a more appropriate observation model, but rather on blindly enlarging the number of degrees of freedom of the orthographic projection model. Thus, although the trajectories are better approximated by the rank 3 factorization of \mathbf{R} than the rank 1 factorization of $\tilde{\mathbf{R}}$, the estimates of the 3D structure are not more accurate.

8.7 Summary

In this chapter we develop an efficient method to recover three-dimensional (3D) rigid structure from a set of trajectories of projections of feature points.

We exploit linear subspace constraints to derive a solution for the *structure from motion* (SFM) problem that is based on the factorization of a matrix $\tilde{\mathbf{R}}$ that is rank 1 in a noiseless situation, rather than a rank 3 matrix \mathbf{R} as in the original factorization method of Tomasi and Kanade [59, 61]. The rank 1 matrix $\tilde{\mathbf{R}}$ is obtained from the measurement matrix \mathbf{R} by making an adequate subspace projection.

Our algorithm is computationally simple because we use a fast iterative method to compute the rank 1 matrix that best matches the data, avoiding the need to compute the *Singular Value Decomposition* (SVD) of the measurement matrix.

The chapter ends with experimental results that illustrate the properties of the rank 1 matrix $\tilde{\mathbf{R}}$ and compare the computational cost of our approach with the cost of the original factorization method of Tomasi and Kanade [59, 61]. We conclude that the *rank 1 factorization* requires a number of floating point operations (FLOPS) dramatically smaller than the one required by the original factorization method. The computational gain factor is approximately 20 when processing 50 frames and 50 feature points, and even larger when processing a larger number of features and/or a large number of frames. The use of an iterative algorithm, rather than the SVD, in the rank 3 factorization also proves to be slower than the rank 1 factorization by a factor of approximately 2.

Chapter 9

Surface-Based Factorization

9.1 Introduction

Feature tracking may be unreliable when processing noisy video sequences. To overcome this difficulty, the factorization approach described in this chapter extends the feature-based factorization method described in chapter 8 in the following sense. Instead of tracking point-wise features, we track larger regions where the image motion is described by a single set of parameters. We show how to recover three-dimensional (3D) structure (3D motion and a parametric description of the 3D shape) from the set of parameters describing the image motion by using a factorization approach.

In this chapter we represent the 3D shape by a parametric description of the object surface. To emphasize that the description is surface-based, we call our approach the *surface-based factorization method*. The parametric description of the 3D shape induces a parameterization for the two-dimensional (2D) motion of the brightness pattern on the image plane. The 2D motion parameters are estimated by using the method described in chapter 3. In this chapter we show how to recover the parameters describing the 3D shape and the 3D motion from the parameters describing the 2D motion on the image plane. This is done by factorizing a surface-based measurement matrix whose entries are the parameters describing the image motion. To compute suitable factors of this measurement matrix, we use the *rank 1 factorization* procedure introduced in chapter 8.

The surface-based factorization method includes the feature-based factorization

method as a special case. In fact, the feature-based approach describes the 3D shape by a sparse set of small regions with relative depth constant in each region. This description corresponds to parameterizing the 3D shape with one parameter per region within the surface-based factorization approach. These parameters code the relative depth of the corresponding region (called feature in this case).

In this chapter we also show how to include confidence weights for the parameters describing the 2D motion in the image plane (or for the feature trajectories). The accuracy of the estimates of the 2D motion parameters depends on the brightness pattern of the region considered, as well as on its size, as studied in chapter 3. By taking into account the confidence weights, we weight differently the 2D motion parameter estimates corresponding to different regions. For example, we weight more the trajectory of a “sharp” feature than the trajectory of a “smooth” feature. Reference [47] proposes an iterative method to solve a more general problem, where the weights are time-variant. As reported in reference [47] the iterative method may fail to converge. We show that when the weights are time-invariant, the problem is rewritten as the non-weighted factorization of a modified matrix. Then, any method can be used to factorize this matrix. We call this extension the *weighted factorization method*.

The chapter is organized as follows. In section 9.2 we introduce our approach by using an illustrative example. In section 9.3 we describe the surface-based factorization method with full generality. In section 9.4, we show how to accommodate confidence weights for the feature trajectories, or for the parameters describing the image motion. Section 9.5 describes two experiments that illustrate the rank 1 surface-based factorization and the weighted factorization. Section 9.6 summarizes the content of the chapter.

9.2 Example

The surface-based factorization method uses a parametric description of the surface \mathcal{S} of the rigid object in terms of a parameter vector \mathbf{a} , $\mathcal{S}(\mathbf{a})$. We exploit the constraints induced

on the two-dimensional (2D) motion in the image plane by the projection operator, the rigidity of the object, and the parameterization of the surface shape of the object. The constraints induced on the image motion enable us to parameterize the image motion mapping $\mathbf{u}_f(\mathbf{s})$ (this mapping was introduced in chapter 7) in terms of a parameter vector \mathbf{d}_f as $\mathbf{u}(\mathbf{d}_f; \mathbf{s})$. The parameter vector \mathbf{d}_f is directly related to the three-dimensional (3D) shape parameter vector \mathbf{a} and the 3D position \mathbf{m}_f , as will be shown below. Our approach follows these two stages. First, we estimate the parameters $\{\mathbf{d}_f\}$ by using the numerical technique for image motion estimation described in chapter 3. Then, we solve the inverse problem of going from the sequence of image motion parameters to the 3D structure, i.e., we determine the 3D shape parameter vector \mathbf{a} and the sequence of 3D positions $\{\mathbf{m}_f\}$, given the estimates $\{\hat{\mathbf{d}}_f\}$ of the image motion parameter vectors $\{\mathbf{d}_f\}$.

Before addressing the general case, we illustrate our approach with a simple example: a parabolic patch moving in a 2D world where the images are 1D orthogonal projections. This scenario, although simpler than the 3D world problem, reflects the very basic properties and difficulties of the *structure from motion* (SFM) paradigm. Note that the 2D scenario, illustrated in Figure 9.1, corresponds to the real 3D world, if we consider only one epipolar plane and assume that the motion occurs on that plane. The images are single scan-lines. The 2D world was also the scenario used by Tomasi and Kanade to introduce the *factorization method* in reference [60].

Figure 9.1 shows a parabolic patch \mathcal{S} that moves with respect to a fixed camera. We attach a coordinate system to the object \mathcal{S} given by the axes labeled by x and z . The 2D object shape $\mathcal{S}(\mathbf{a})$ is described in terms of the parameter vector $\mathbf{a} = [a_0, a_1, a_2]^T$, in the object coordinate system, by the parabola

$$z(x) = z(\mathbf{a}; x) = a_0 + a_1x + a_2x^2. \quad (9.1)$$

To capture the motion of the object, we attach a different coordinate system to the camera given by the axes u and w , see Figure 9.1. The u axis is the camera “plane”. We define the 2D motion of the object by specifying the position of the object coordinate

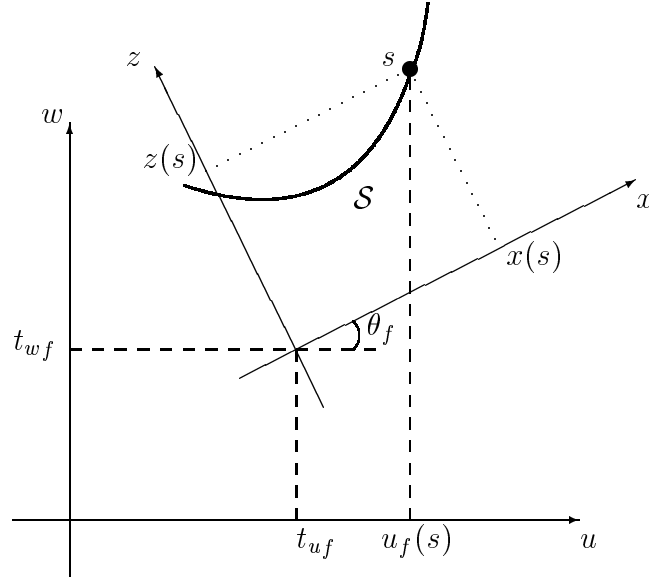


Figure 9.1: 2D world: object and camera coordinate systems.

system relative to the camera coordinate system. The unconstrained motion of a rigid body can be described in terms of a time varying point translation and a rotation. Hence, the object position at time instant f is expressed in terms of $(t_{uf}, t_{wf}, \theta_f)$ where, see Figure 9.1, (t_{uf}, t_{wf}) are chosen to be the coordinates of the origin of the object coordinate system with respect to the camera coordinate system (translational component of the 2D motion), and θ_f is the orientation of the object coordinate system relative to the camera coordinate system (rotational component of the motion).

At instant f , the point on the object with 2D coordinates (x, z) in the object coordinate system has the following coordinates in the camera coordinate system, see Figure 9.1,

$$\begin{bmatrix} u_f \\ w_f \end{bmatrix} = \Theta_f \begin{bmatrix} x \\ z \end{bmatrix} + \mathbf{t}_f = \begin{bmatrix} i_{xf} & i_{zf} \\ k_{xf} & k_{zf} \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} t_{uf} \\ t_{wf} \end{bmatrix}, \quad (9.2)$$

where Θ_f is the rotation matrix and \mathbf{t}_f is the translation vector. The elements of the rotation matrix Θ_f are $i_{xf} = \cos \theta_f$, $i_{zf} = -\sin \theta_f$, $k_{xf} = \sin \theta_f$, and $k_{zf} = \cos \theta_f$, where θ_f is the angle indicated in Figure 9.1.

From expression (9.2), we see that the point (x, z) projects at time f on the image

coordinate u_f given by

$$u_f = i_{xf}x + i_{zf}z + t_{uf}. \quad (9.3)$$

Expression (9.3) shows that the orthogonal projection is insensitive to the translation component t_{wf} of the object motion. This reflects the well known fact that, under orthography, the absolute depth (distance from the camera to the object) cannot be estimated. Only the set of positions $\{\mathbf{m}_f = \{t_{uf}, \theta_f\}, 1 \leq f \leq F\}$ can be estimated from the image sequence.

We now show for this example how the mapping $\mathbf{u}_f(\mathbf{s})$, introduced in chapter 7 and illustrated in Figure 7.1, is described parametrically. In the 2D world of this example, the mapping $\mathbf{u}_f(\mathbf{s})$ is written as $u_f(s)$ because it maps a scalar s to a scalar u . Choose the coordinate s , labeling the argument of the texture function \mathcal{T} , and representing in a unique way the generic point on the object surface (object contour in this case), to be the object coordinate x . We refer to s as the texture coordinate. A point with texture coordinate s on the object surface projects at time f , according to expression (9.3), to the image coordinate $u_f(s)$ given by

$$\begin{aligned} u_f(s) &= i_{xf}x(s) + i_{zf}z(s) + t_{uf} \\ &= i_{xf}s + i_{zf}(a_0 + a_1s + a_2s^2) + t_{uf}, \end{aligned} \quad (9.4)$$

where $x(s)$ and $z(s)$ are the coordinates of the point s in the object coordinate system. The equality $x(s) = s$ comes from the choice of the texture coordinate s , and the expression for $z(s)$ comes from the parabolic shape (9.1).

By defining the coefficients of the powers of s in expression (9.4) as

$$\begin{cases} d_{f0} = i_{zf}a_0 + t_{uf} \\ d_{f1} = i_{xf} + i_{zf}a_1 \\ d_{f2} = i_{zf}a_2, \end{cases} \quad (9.5)$$

we have the following parametric description for the image motion $u_f(s)$ in terms of the parameter vector $\mathbf{d}_f = [d_{f0}, d_{f1}, d_{f2}]^T$,

$$u_f(s) = u(\mathbf{d}_f; s) = d_{f0} + d_{f1}s + d_{f2}s^2. \quad (9.6)$$

The parameter vector $\mathbf{d}_f = [d_{f0}, d_{f1}, d_{f2}]^T$ describes the motion of the brightness pattern in the image plane, i.e., it describes the mapping $\mathbf{u}_f(\mathbf{s})$ introduced in chapter 7, see Figure 7.1.

With the parabolic patch, the steps of our approach to recover the 2D structure, i.e., the shape parameters $\{a_0, a_1, a_2\}$ and the set of positions $\{t_{uf}, \theta_f, 1 \leq f \leq F\}$ are then summarized as:

- (i) Given the image sequence of F frames, estimate the set of image motion parameters $\{d_{f0}, d_{f1}, d_{f2}, 1 \leq f \leq F\}$. This leads to $3F$ estimates $\{\hat{d}_{f0}, \hat{d}_{f1}, \hat{d}_{f2}, 1 \leq f \leq F\}$.
- (ii) Invert the relation (9.5), solving for the shape parameters $\{a_0, a_1, a_2\}$ and the $2F$ object positions $\{t_{uf}, \theta_f, 1 \leq f \leq F\}$, given the set of $3F$ estimates $\{\hat{d}_{f0}, \hat{d}_{f1}, \hat{d}_{f2}\}$.

The step (i) above is solved by using the numerical technique described in chapter 3 to fit parametric models to the motion of the brightness patterns in the image plane. The step (ii) above leads in general to a nonlinear problem. Section 9.3 details our approach to this problem. First, we obtain a closed-form solution for the estimate of the 3D translation. Then, due to the structure of the orthogonal projection operator, and the shape parameterization, we can express the dependence of $\mathbf{d}_f = \mathbf{d}(\mathbf{a}, \mathbf{m}_f)$ for $1 \leq f \leq F$ on the vectors \mathbf{a} and \mathbf{m}_f in a bilinear matrix format as $\mathbf{R} = \mathbf{M}\mathbf{S}^T$, where the matrix \mathbf{R} collects the image motion parameters $\{\mathbf{d}_f, 1 \leq f \leq F\}$, \mathbf{M} depends on the positions $\{\mathbf{m}_f, 1 \leq f \leq F\}$, and \mathbf{S} contains the shape parameter \mathbf{a} . The problem of estimating \mathbf{a} and $\{\mathbf{m}_f, 1 \leq f \leq F\}$ becomes how to find suitable factors \mathbf{M} and \mathbf{S}^T for the factorization of the matrix \mathbf{R} . We will see that the *rank 1 factorization* procedure introduced in chapter 8 solves this problem.

Our general methodology can be used for any parametric shape description. The situations we are interested in are characterized by no prior knowledge about the object shape. For this kind of situations, a general shape model must be characterized by a local parameterization. The local shape parameterization induces a local parameterization for the motion in the image plane. In the following section we detail our approach for a

generic shape model locally parameterized: the piecewise polynomial functions.

9.3 Surface-Based Factorization

We consider a rigid body moving in front of the camera. The scenario was introduced in chapter 8 and it is illustrated in Figure 8.1. We attach coordinate systems to the object and to the camera. The object coordinate system (o.c.s.) has axes labeled by x, y , and z . The camera coordinate system (c.c.s.) has axes labeled by u, v , and w . We consider that the o.c.s. coincides with the c.c.s. on the first frame. The image plane is defined by the axes u and v . The images are modeled as orthographic projections of the object texture. Our algorithm is easily extended to the scaled-orthography and the paraperspective projections by proceeding as references [47, 48, 49] propose for the original factorization method of Tomasi and Kanade [59, 60, 61].

3D shape

The three-dimensional (3D) shape of the rigid object is a parametric description of the object surface. We consider objects whose shape is given by a piecewise polynomial surface with N patches. The 3D shape is described in terms of N sets of parameters $\{a_{00}^n, a_{10}^n, a_{01}^n, a_{11}^n, a_{20}^n, a_{02}^n, \dots\}$, for $1 \leq n \leq N$, where

$$\begin{aligned} z = & a_{00}^n + a_{10}^n(x - x_0^n) + a_{01}^n(y - y_0^n) \\ & + a_{11}^n(x - x_0^n)(y - y_0^n) + a_{20}^n(x - x_0^n)^2 + a_{02}^n(y - y_0^n)^2 \\ & + \dots \end{aligned} \tag{9.7}$$

describes the shape of the patch n in the o.c.s. With respect to the representation of the polynomial patches, the parameters x_0^n and y_0^n can have any value, for example they can be made zero. We allow the specification of general parameters x_0^n, y_0^n because the shape of a small patch n with support region $\{(x, y)\}$ located far from the point (x_0^n, y_0^n) has a high sensitivity with respect to the shape parameters. To minimize this sensitivity, we choose for (x_0^n, y_0^n) the centroid of the support region of patch n . With this choice, we

improve the accuracy of the 3D structure recovery algorithm. As we will see below, by making (x_0^n, y_0^n) to be the centroid of the support region of patch n , we also improve the numerical stability of the algorithm that estimates the two-dimensional (2D) motion in the image plane.

Expression (9.7) describes the 3D shape with full generality in a local way – it is the Taylor series expansion of the relative depth $z(x, y)$ around the point $(x, y) = (x_0^n, y_0^n)$, for appropriate values of the set of shape parameters $\{a_{00}^n, a_{10}^n, a_{01}^n, a_{11}^n, a_{20}^n, a_{02}^n, \dots\}$. We can recover the simpler feature-based shape description from the general 3D shape described by expression (9.7) by making zero all the shape parameters, except for a_{00}^n that codes the relative depth of feature n , $z = a_{00}^n$. Expression (9.7) models also a special case of practical interest: the piecewise planar shapes. In this case, the planar patch n is described by the parameters $\{a_{00}^n, a_{10}^n, a_{01}^n\}$. This set of parameters codes the *orientation* of the planar patch, besides its position.

Since the following derivations deal with a single surface patch, we will omit the super-index n in the shape parameters. To further simplify the notation, we define the vectors $\mathbf{a}_1 = [a_{10}, a_{01}]^T$, $\mathbf{s} = [x, y]^T$, $\mathbf{s}_0 = [x_0, y_0]^T$, $\mathbf{a}_2 = [a_{11}, a_{20}, a_{02}, \dots]^T$, and $\mathbf{p}(\mathbf{s} - \mathbf{s}_0) = [(x - x_0)(y - y_0), (x - x_0)^2, (y - y_0)^2, \dots]^T$, and rewrite the shape of patch k as

$$z = a_{00} + \mathbf{a}_1^T(\mathbf{s} - \mathbf{s}_0) + \mathbf{a}_2^T \mathbf{p}(\mathbf{s} - \mathbf{s}_0). \quad (9.8)$$

The vectors \mathbf{a}_2 and $\mathbf{p}(\mathbf{s} - \mathbf{s}_0)$ are $P \times 1$ where P depends on the degree of the polynomial surface patch.

3D motion

As detailed in chapter 8, we define the 3D motion of the object by specifying the position of the o.c.s. relative to the c.c.s. in terms of $(t_{uf}, t_{vf}, t_{wf}, \Theta_f)$ where (t_{uf}, t_{vf}, t_{wf}) are the coordinates of the origin of the o.c.s. with respect to the c.c.s. (3D translation), and the matrix Θ_f is the rotation matrix that determines the orientation of the o.c.s. relative to the c.c.s. (3D rotation). A point with coordinates $[x, y, z]^T$ in the o.c.s. has the following

coordinates in the c.c.s., at frame f ,

$$\begin{bmatrix} u_f \\ v_f \\ w_f \end{bmatrix} = \Theta_f \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_{uf} \\ t_{vf} \\ t_{wf} \end{bmatrix}, \quad (9.9)$$

where the 3D rotation matrix Θ_f is expressed in terms of the Euler angles as in expression (8.1).

Under orthography, the point with coordinates $[x, y, z]^T$ in the o.c.s. projects in frame f onto the image point $[u_f, v_f]^T$ given by

$$\begin{bmatrix} u_f \\ v_f \end{bmatrix} = \mathbf{M}_f \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{t}_f, \quad (9.10)$$

where the matrix \mathbf{M}_f collects the first and second rows of the 3D rotation matrix Θ_f and the vector \mathbf{t}_f contains the two components of the 3D translation that can be recovered from the image sequence, $\mathbf{t}_f = [t_{uf}, t_{vf}]^T$.

We now show that the 2D motion induced in the image plane by the body-camera 3D motion is described in terms of a set of parameters. For example, for planar surface patches the 2D motion in the image plane is given by the affine motion model studied in chapter 3. We relate the parameters of the 2D motion model to the 3D shape and 3D motion parameters.

Image motion

Consider a generic point in the object surface with coordinates $\mathbf{s} = [x, y]^T$ and z given by expression (9.8). We denote by $\mathbf{u}_f(\mathbf{s}) = [u_f(\mathbf{s}), v_f(\mathbf{s})]^T$ the trajectory of the projection of the point \mathbf{s} in the image plane. Since we have chosen the coordinate systems to coincide on the first frame, we have $\mathbf{u}_1(\mathbf{s}) = \mathbf{s}$. At frame f , the point \mathbf{s} projects according to expression (9.10), to the image point

$$\mathbf{u}_f(\mathbf{s}) = \mathbf{N}_f \mathbf{s} + \mathbf{n}_f z + \mathbf{t}_f, \quad (9.11)$$

where we have decomposed the matrix \mathbf{M}_f as $\mathbf{M}_f = [\mathbf{N}_f, \mathbf{n}_f]$ where the 2×2 matrix \mathbf{N}_f collects the first and second columns of the matrix \mathbf{M}_f and the vector \mathbf{n}_f is the third column of \mathbf{M}_f .

By inserting expression (9.8) into expression (9.11), we express the image displacement between frame 1 and frame f in terms of the 3D shape and 3D motion parameters. After simple manipulations, we obtain

$$\mathbf{u}_f(\mathbf{s}) = \mathbf{N}_f \mathbf{s}_0 + \mathbf{n}_f a_{00} + \mathbf{t}_f + (\mathbf{N}_f + \mathbf{n}_f \mathbf{a}_1^T) (\mathbf{s} - \mathbf{s}_0) + \mathbf{n}_f \mathbf{a}_2^T \mathbf{p}(\mathbf{s} - \mathbf{s}_0). \quad (9.12)$$

Denoting the 2×1 vector corresponding to the term independent of \mathbf{s} , the 2×2 matrix that multiplies $(\mathbf{s} - \mathbf{s}_0)$, and the $2 \times P$ matrix that multiplies $\mathbf{p}(\mathbf{s} - \mathbf{s}_0)$ by

$$\begin{cases} \mathbf{d}_f = \mathbf{N}_f \mathbf{s}_0 + \mathbf{n}_f a_{00} + \mathbf{t}_f \\ \mathbf{D}_f = \mathbf{N}_f + \mathbf{n}_f \mathbf{a}_1^T \\ \mathbf{E}_f = \mathbf{n}_f \mathbf{a}_2^T, \end{cases} \quad (9.13)$$

we rewrite expression (9.12) as

$$\mathbf{u}_f(\mathbf{s}) = \mathbf{d}_f + \mathbf{D}_f (\mathbf{s} - \mathbf{s}_0) + \mathbf{E}_f \mathbf{p}(\mathbf{s} - \mathbf{s}_0). \quad (9.14)$$

Expression (9.14) shows that the image coordinates at frame f , \mathbf{u}_f , of the points belonging to the object surface are parametric mappings of their image coordinates in frame 1, $\mathbf{u}_1 = \mathbf{s}$. The 2D motion of the brightness pattern in the image plane is then described parametrically by expression (9.14). Expression (9.13) relates the parameters of the 2D motion model for each surface patch, \mathbf{d}_f , \mathbf{D}_f , and \mathbf{E}_f , to the 3D motion parameters, \mathbf{N}_f , \mathbf{n}_f , and \mathbf{t}_f , and the 3D shape parameters corresponding to that patch, a_{00} , \mathbf{a}_1 , and \mathbf{a}_2 .

For the special case of piecewise planar surfaces, the 3D shape of each patch is described by a_{00} and \mathbf{a}_1 and the 2D motion in the image plane is given by $\mathbf{u}_f(\mathbf{s}) = \mathbf{d}_f + \mathbf{D}_f (\mathbf{s} - \mathbf{s}_0)$, i.e., it is the affine motion model studied in chapter 3. As we saw in section 3.5, the choice for $\mathbf{s}_0 = [x_0, y_0]^T$ as the centroid of the support region of the surface patch improves the numerical stability of the image motion estimation algorithm.

Except for particular 3D motions, the 2D motion in the image plane corresponding to different surface patches is described by different model parameterizations. The problem of estimating the support regions of the surface patches leads to the segmentation of the image motion field. Segmentation from motion has been widely addressed in the past. In section 3.1 we overviewed the existing methods. In our experiments, we used two methods

that lead to similar results. The first method simply slides a rectangular window across the image and detects abrupt changes in the motion parameters. The second method uses a quad-tree decomposition. We start by estimating the motion parameters considering the entire image as the support region. The region is recursively decomposed into smaller regions and the motion of each sub-region is estimated. Then we associate regions with similar motion. Another possible way to use our surface-based factorization method is to select *a priori* the support regions of the surface patches. This is the approach followed by the feature-based methods, where the features are selected *a priori*, based on the spatial variability of the brightness pattern, rather than on any 2D motion cue.

3D structure from 2D motion

The problem of inferring 3D rigid structure from the image motion is formulated as estimating the 3D motion parameters $\{\mathbf{N}_f, \mathbf{n}_f, \mathbf{t}_f, 2 \leq f \leq F\}$ and the 3D shape parameters $\{a_{00}^n, \mathbf{a}_1^n, \mathbf{a}_2^n, 1 \leq n \leq N\}$ from the image motion parameters $\{\mathbf{d}_f^n, \mathbf{D}_f^n, \mathbf{E}_f^n, 2 \leq f \leq F, 1 \leq n \leq N\}$ by inverting the overconstrained set of equations of expression (9.13) for all the frames and all the surface patches. The super-index n above denotes the surface patch.

We start by estimating the translation vector \mathbf{t}_f . By choosing the o.c.s. in such a way that $\sum_n a_{00}^n = 0$, and the image origin in such a way that $\sum_n \mathbf{s}_0^n = [0, 0]^T$, we obtain the Least Squares (LS) estimate for the translation vector \mathbf{t}_f as the mean of the vectors $\{\mathbf{d}_f^n, 1 \leq n \leq N\}$,

$$\hat{\mathbf{t}}_f = \frac{1}{N} \sum_{n=1}^N \mathbf{d}_f^n. \quad (9.15)$$

To eliminate the dependence of the image motion parameters on the translation, we replace the translation estimates into expression (9.13) and define a new set of parameters $\{\tilde{\mathbf{d}}_f^n\}$ related to $\{\mathbf{d}_f^n\}$ by

$$\tilde{\mathbf{d}}_f^n = \mathbf{d}_f^n - \frac{1}{N} \sum_{m=1}^N \mathbf{d}_f^m. \quad (9.16)$$

Defining the $2 \times (P + 3)$ matrix \mathbf{R}_f and the $3 \times (P + 3)$ matrix \mathbf{S}^T as

$$\mathbf{R}_f = \begin{bmatrix} \tilde{\mathbf{d}}_f & \mathbf{D}_f & \mathbf{E}_f \end{bmatrix} \quad \text{and} \quad \mathbf{S}^T = \begin{bmatrix} \mathbf{s}_0 & \mathbf{I}_{2 \times 2} & \mathbf{0}_{2 \times P} \\ a_{00} & \mathbf{a}_1^T & \mathbf{a}_2^T \end{bmatrix}, \quad (9.17)$$

we rewrite the equation system (9.13) in matrix format as

$$\mathbf{R}_f^n = \mathbf{M}_f \mathbf{S}_n^T, \quad (9.18)$$

where the 2×3 matrix \mathbf{M}_f contains the two first rows of the 3D rotation matrix Θ_f , and the index n in \mathbf{R}_f^n and \mathbf{S}_n^T denotes the surface patch number. Expression (9.18) relates the image motion parameters at frame f and patch n , in matrix \mathbf{R}_f^n , to the 3D rotation at frame f , in matrix \mathbf{M}_f , and the 3D shape parameters for the patch n , in matrix \mathbf{S}_n^T .

Surface-based factorization

There are $N(F-1)$ matrix equations like expression (9.18): one for each surface patch $1 \leq n \leq N$ and each frame $2 \leq f \leq F$. To make explicit the entire set of equations that arise from considering every patch and every frame, we define the $2(F-1) \times N(P+3)$ matrix \mathbf{R} of image motion parameters, the $2(F-1) \times 3$ matrix \mathbf{M} of 3D rotation parameters, and the $3 \times N(P+3)$ matrix \mathbf{S}^T of 3D shape parameters as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{R}_2^2 & \cdots & \mathbf{R}_2^N \\ \mathbf{R}_3^1 & \mathbf{R}_3^2 & \cdots & \mathbf{R}_3^N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_F^1 & \mathbf{R}_F^2 & \cdots & \mathbf{R}_F^N \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_2 \\ \mathbf{M}_3 \\ \vdots \\ \mathbf{M}_F \end{bmatrix}, \quad \mathbf{S}^T = [\mathbf{S}_1^T \quad \mathbf{S}_2^T \quad \cdots \quad \mathbf{S}_N^T], \quad (9.19)$$

and write the relation between the image motion parameters and the 3D structure parameters as

$$\mathbf{R} = \mathbf{M} \mathbf{S}^T. \quad (9.20)$$

Expression (9.20) is an extension of the expression (8.10) introduced in chapter 8 and first derived by Tomasi and Kanade in references [59, 60, 61]. Expression (9.20), unlike (8.10), accommodates regions whose shape is parameterized rather than described by a single point. In fact, the measurement matrix involved in the feature-based approach described in chapter 8 is composed by the columns of the matrix \mathbf{R} in expressions (9.19) and (9.20) that contain the parameters $\{\tilde{\mathbf{d}}_f^n, 1 \leq n \leq N, 2 \leq f \leq F\}$,

see expressions (9.19) and (9.17). For this reason, we call the matrix \mathbf{R} in expressions (9.19) and (9.20) the *surface-based measurement matrix*. The shape matrix involved in the feature-based factorization is composed by the columns of the matrix \mathbf{S}^T in expressions (9.19) and (9.20) that contain the parameters $\{\mathbf{s}_0^n, a_{00}^n, 1 \leq n \leq N\}$, see expressions (9.19) and (9.17). The motion matrix \mathbf{M} in expressions (9.19) and (9.20) is the same matrix that appears in the feature-based factorization.

Expression (9.20) shows that the surface-based measurement matrix \mathbf{R} of the image motion parameters is rank deficient. In a noiseless situation, the surface-based measurement matrix \mathbf{R} is rank 3 reflecting the high redundancy in the image motion parameters, due to the rigidity of the object. Thus, the surface-based measurement matrix \mathbf{R} has the same rank of the measurement matrix involved in the feature-based factorization of chapter 8. The problem of estimating the 3D shape and 3D motion parameters amounts to finding suitable factors of the surface-based measurement matrix \mathbf{R} . The 3D shape matrix \mathbf{S} and the 3D motion matrix \mathbf{M} are then the solution of

$$\min_{\mathbf{M}, \mathbf{S}} \|\mathbf{R} - \mathbf{M}\mathbf{S}^T\|_F, \quad (9.21)$$

where the rows of the matrices \mathbf{M} and \mathbf{S}^T are restricted to have the special structure of expression (9.19) – the rows of \mathbf{M} are restricted to have: i) unit norm, and ii) row i orthogonal to row $i + 1$; and the first two rows of \mathbf{S}^T are given by expression (9.17).

To factorize the surface-based measurement matrix \mathbf{R} , i.e., to solve the minimization (9.21), we use the *rank 1 factorization method* described in chapter 8, section 8.3. Note that the 3D motion matrix \mathbf{M} is the same matrix involved in the feature-based factorization of chapter 8, and the first two rows of the 3D shape matrix \mathbf{S}^T in expression (9.21) are known from expressions (9.19) and (9.17), as required by our rank 1 factorization algorithm.

For the special case of piecewise planar shapes, the submatrices \mathbf{R}_f^n and \mathbf{S}_n^T in expressions (9.17) and (9.18) are simplified – they have only the first 3 columns. In this case, the surface-based measurement matrix \mathbf{R} is $2(F - 1) \times 3N$ and the shape matrix \mathbf{S}^T

is $3 \times 3N$.

9.4 Weighted Factorization

The accuracy of the estimates of the two-dimensional (2D) motion parameters depends on the spatial variability of the brightness intensity pattern and on the size of the image region considered, as studied in chapter 3. The factorization method proposed in the previous section, as well as the feature-based factorization of chapter 8 and the original method of Tomasi and Kanade [59, 60, 61], weights equally the contribution of each region, or feature, to the final three-dimensional (3D) shape and 3D motion estimates. A more robust estimate weights more heavily a trajectory corresponding to a “sharp” feature than a trajectory corresponding to a feature with a more smooth texture. In this section, we introduce such a model and show that it leads to the factorization of a modified measurement matrix. The computation of the weighted estimates is then done by using the *rank 1 factorization method* introduced in section 8.3, i.e., without additional computational cost.

We consider the model in expressions (9.13). Each 2D motion parameter of the surface patch n , in vector \mathbf{d}_f^n and matrices \mathbf{D}_f^n and \mathbf{E}_f^n , is observed with additive Gaussian white noise. The noise variance for each of the two components of the vector \mathbf{d}_f^n is denoted by $(\sigma_d^n)^2$, the noise variance for the two entries of the first column of the matrix \mathbf{D}_f^n is denoted by $(\sigma_{D1}^n)^2$, the noise variance for the two entries of the second column of the matrix \mathbf{D}_f^n is denoted by $(\sigma_{D2}^n)^2$, the noise variance for the two entries of the column i of the matrix \mathbf{E}_f^n is denoted by $(\sigma_{Ei}^n)^2$. We impose the variance of the two components of each column to be the same so that the weighted problem can still be solved by the rank 1 factorization method, except now of a modified measurement matrix, as it will become clear below.

The variances $(\sigma_d^n)^2$, $(\sigma_{Di}^n)^2$, and $(\sigma_{Ei}^n)^2$ of the errors of the 2D motion parameters are estimated from the spatial gradient of the image brightness pattern as detailed in

chapter 3. For the feature-based case, the 2D motion is described by the trajectories of the features, i.e., by the parameters $\{\mathbf{d}_f^n, 1 \leq n \leq N, 2 \leq f \leq F\}$, thus the error variances are $\{(\sigma_d^n)^2, 1 \leq n \leq N\}$. These variances are computed as described in section 3.4, i.e., by evaluating expression (3.44) with $\sigma_t = 1$. For the piecewise planar shapes scenario, the 2D motion is described by the set of affine mappings with parameters $\{\mathbf{d}_f^n, \mathbf{D}_f^n, 1 \leq n \leq N, 2 \leq f \leq F\}$, thus the error variances are $\{(\sigma_d^n)^2, (\sigma_{D1}^n)^2, (\sigma_{D2}^n)^2, 1 \leq n \leq N\}$. Each of these variances is computed as described in section 3.5, i.e., by evaluating expression (3.60) where the set \mathcal{P} is the corresponding subset of the parameters of the affine motion model and $\sigma_t = 1$.

We follow the same strategy of the previous section: first, estimate the translation, then replace the translation estimates and solve for the 3D rotation and 3D shape. By choosing the origin of the o.c.s. in such a way that $\sum_n \mathbf{s}_0^n / (\sigma_d^n)^2 = [0, 0]^T$ and $\sum_n a_{00}^n / (\sigma_d^n)^2 = 0$, we get the estimate

$$\hat{\mathbf{t}}_f = \frac{\sum_{n=1}^N \frac{\mathbf{d}_f^n}{(\sigma_d^n)^2}}{\sum_{n=1}^N \frac{1}{(\sigma_d^n)^2}} \quad (9.22)$$

for the translation along the camera plane. Expression (9.22) generalizes expression (9.15). Replacing the translation estimates in equation system (9.13), and defining the set of parameters $\{\tilde{\mathbf{d}}_f^n\}$ related to $\{\mathbf{d}_f^n\}$ by

$$\tilde{\mathbf{d}}_f^n = \mathbf{d}_f^n - \frac{\sum_{m=1}^N \frac{\mathbf{d}_f^m}{(\sigma_d^m)^2}}{\sum_{m=1}^N \frac{1}{(\sigma_d^m)^2}}, \quad (9.23)$$

and the matrices \mathbf{R} , \mathbf{M} , and \mathbf{S} as in expressions (9.17) and (9.19), we obtain, as before,

$$\mathbf{R} = \mathbf{M}\mathbf{S}^T. \quad (9.24)$$

To take into account the different variances of the errors of the entries of the measurement matrix \mathbf{R} , we define the $2(F-1) \times N(P+3)$ weight matrix \mathbf{W} as

$$\mathbf{W} = \begin{bmatrix} \frac{1}{\sigma_d^1} & \frac{1}{\sigma_{D1}^1} & \frac{1}{\sigma_{D2}^1} & \frac{1}{\sigma_{E1}^1} & \cdots & \frac{1}{\sigma_{EP}^1} & \cdots & \cdots & \frac{1}{\sigma_d^N} & \frac{1}{\sigma_{D1}^N} & \frac{1}{\sigma_{D2}^N} & \frac{1}{\sigma_{E1}^N} & \cdots & \frac{1}{\sigma_{EP}^N} \\ \frac{1}{\sigma_d^1} & \frac{1}{\sigma_{D1}^1} & \frac{1}{\sigma_{D2}^1} & \frac{1}{\sigma_{E1}^1} & \cdots & \frac{1}{\sigma_{EP}^1} & \cdots & \cdots & \frac{1}{\sigma_d^N} & \frac{1}{\sigma_{D1}^N} & \frac{1}{\sigma_{D2}^N} & \frac{1}{\sigma_{E1}^N} & \cdots & \frac{1}{\sigma_{EP}^N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\sigma_d^1} & \frac{1}{\sigma_{D1}^1} & \frac{1}{\sigma_{D2}^1} & \frac{1}{\sigma_{E1}^1} & \cdots & \frac{1}{\sigma_{EP}^1} & \cdots & \cdots & \frac{1}{\sigma_d^N} & \frac{1}{\sigma_{D1}^N} & \frac{1}{\sigma_{D2}^N} & \frac{1}{\sigma_{E1}^N} & \cdots & \frac{1}{\sigma_{EP}^N} \end{bmatrix} \quad (9.25)$$

where entry (i, j) of the weight matrix \mathbf{W} represents the weight to be given to entry (i, j) of matrix \mathbf{R} . We formulate the weighted factorization in terms of the *Maximum Likelihood* (ML) estimation. The ML estimate generalizes (9.21) as

$$\min_{\mathbf{M}, \mathbf{S}} \|(\mathbf{R} - \mathbf{MS}^T) \odot \mathbf{W}\|_F \quad (9.26)$$

where the matrices \mathbf{M} and \mathbf{S} are constrained to have the special structure of the motion and shape matrices in the surface-based factorization of the previous section. The symbol \odot denotes the elementwise product of two matrices, also known as *Hadamard matrix product*, see reference [15].

Due to the structure of \mathbf{W} , we rewrite the minimization in expression (9.26) as the factorization of a modified matrix \mathbf{R}_w ,

$$\min_{\mathbf{M}, \mathbf{S}_w} \|\mathbf{R}_w - \mathbf{MS}_w^T\|_F, \quad (9.27)$$

where the matrices \mathbf{R}_w and \mathbf{S}_w are related to the matrices \mathbf{R} and \mathbf{S} and the entries of the weight matrix \mathbf{W} by

$$\mathbf{R}_w = \mathbf{R} \operatorname{diag}(\{w_{1i}, 1 \leq i \leq N(P+3)\}), \quad (9.28)$$

$$\mathbf{S}_w = \operatorname{diag}(\{w_{1i}, 1 \leq i \leq N(P+3)\})\mathbf{S}, \quad (9.29)$$

where $\operatorname{diag}(\{w_{1i}, 1 \leq i \leq N(P+3)\})$ denotes a $N(P+3) \times N(P+3)$ diagonal matrix whose entry (i, i) is equal to the entry $(1, i)$ of the weight matrix \mathbf{W} .

The factorization in expression (9.27) is similar to the one studied in chapter 8, section 8.3. Note that the modified measurement matrix \mathbf{R}_w and the first two rows of the matrix \mathbf{S}_w^T are known from expressions (9.28) and (9.29) and the motion matrix \mathbf{M} is the same matrix involved in the rank 1 factorization method of section 8.3. We minimize (9.27) by using the rank 1 factorization procedure described in section 8.3 to compute the factor matrices $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{S}}_w$. To compute the estimate $\widehat{\mathbf{S}}$ of the shape matrix from the matrix $\widehat{\mathbf{S}}_w$ we invert expression (9.29), i.e., we compute

$$\widehat{\mathbf{S}} = \operatorname{diag}\left(\left\{w_{1i}^{-1}, 1 \leq i \leq N(P+3)\right\}\right)\widehat{\mathbf{S}}_w. \quad (9.30)$$

Reference [47] also considers the reliability weights in estimating the matrices \mathbf{M} and \mathbf{S} within the original feature-based factorization method of Tomasi and Kanade [59, 60, 61]. In reference [47], the author allows the weight matrix \mathbf{W} to have a general structure, i.e., one can give each feature a weight that varies along time. The solution is found by an iterative process that, as reported in reference [47], may fail to converge. In our formulation, this is not the case because we restrict the weight matrix \mathbf{W} to have the structure of expression (9.25). For a general matrix \mathbf{W} , it is not possible to write the minimization (9.26) in the form of a factorization such as in expression (9.27). In fact, the unconstrained bilinear problem $\min_{\mathbf{M}, \mathbf{S}} \|(\mathbf{R} - \mathbf{MS}^T) \odot \mathbf{W}\|_F$ has a single global minimum, up to a scale factor, when \mathbf{W} has the rows all equal or the columns all equal. It can be shown that this is not true for a generic matrix \mathbf{W} . In this case, the existence of local minima makes nontrivial the using of iterative numerical techniques.

9.5 Experiments

In this section we describe two experiments that illustrate the methods proposed in this chapter. The first experiment illustrates the *surface-based factorization method* by recovering piecewise linear rigid structure in a two-dimensional (2D) world. The second experiment illustrates the *weighted factorization method* by comparing the three-dimensional (3D) shape and 3D motion estimated from a synthetic set of feature trajectories with the non-weighted estimates.

Surface-based factorization

In Figure 9.2 we show a computer generated 1D image sequence. Time increases from top to bottom. The object shape is shown on Figure 9.3 (superimposed with the final estimate). It is piecewise linear with 4 line patches. The object texture is a smooth function defined over the object contour. The time evolution of the translational and rotational components of the motion are shown respectively in the left and right plots of Figure 9.4, also superimposed with their final estimates. We obtained the image sequence

in Figure 9.2 by projecting the object texture on the image plane and by adding noise.

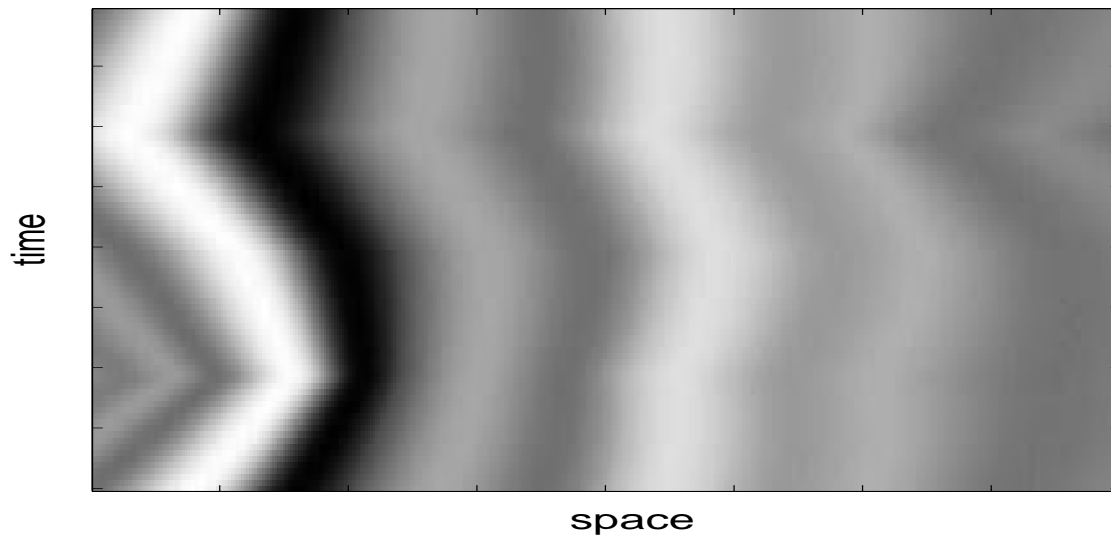


Figure 9.2: 1D image sequence.

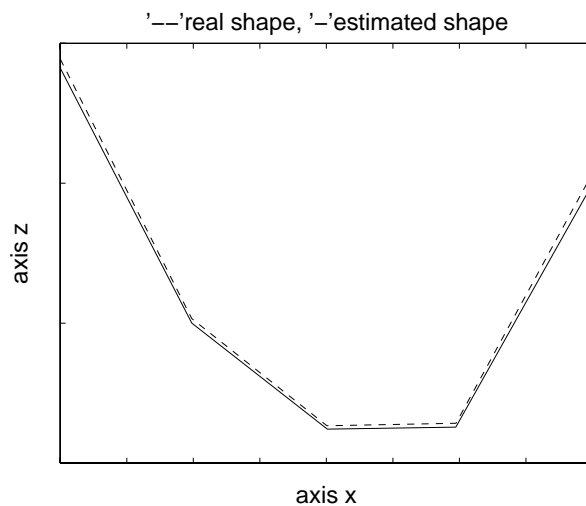


Figure 9.3: 2D shape estimate superimposed with the true 2D shape.

The approach of references [59, 60, 61], as well as the method described in section 4.3, is based on the tracking of feature points. These feature points are selected by using spatial information, in order to make possible their tracking. In general, feature points correspond to brightness corners in 2D images. For 1D images, the feature points would correspond to discontinuities in the brightness pattern. For the sequence represented in

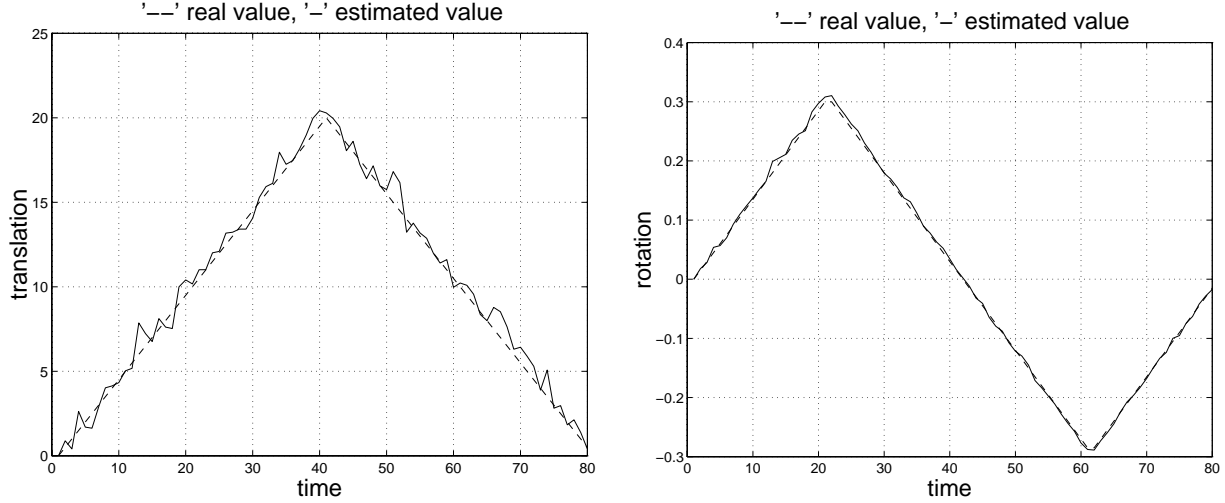


Figure 9.4: 2D motion estimates superimposed with the true values. Left plot: translation along the camera “plane”. Right plot: rotation angle.

Figure 9.2, it is very difficult to select feature points that can be easily tracked because of the smoothness of the brightness pattern. For this reason, the feature-based approaches can not be used to resolve this type of scene, i.e., to infer structure from the motion of smooth textures.

For the linear patch n of the 2D shape, the 1D motion in the image plane is affine with parameters d_{f0}^n and d_{f1}^n . The 1D motion in the image plane is given by the first two terms of expression (9.6) derived for the parabolic patch, i.e., it is described by $\{d_{f0}^n, d_{f1}^n\}$ as $u_f(s) = d_{f0}^n + d_{f1}^n s$. Figure 9.5 shows the estimates for the four affine parameters of the sequence of Figure 9.2. We superimpose the affine parameter estimates with the actual values of the parameters. The graphic on the left shows the evolution of the parameters $\hat{d}_{f0}^1, \hat{d}_{f0}^2, \hat{d}_{f0}^3$, and \hat{d}_{f0}^4 , as functions of the frame index f . The graphic on the right shows the parameters $\hat{d}_{f1}^1, \hat{d}_{f1}^2, \hat{d}_{f1}^3$, and \hat{d}_{f1}^4 . We can see that the variance of the affine parameter estimates is different from one patch to another. This is because the spatial brightness variations and the patch sizes are not the same for the four patches: the smaller the variation of the brightness patch and the smaller the size of a patch are, the higher the variances of the corresponding flow parameter estimates are.

Figures 9.3 and 9.4 show the result of applying our surface-based factorization tech-

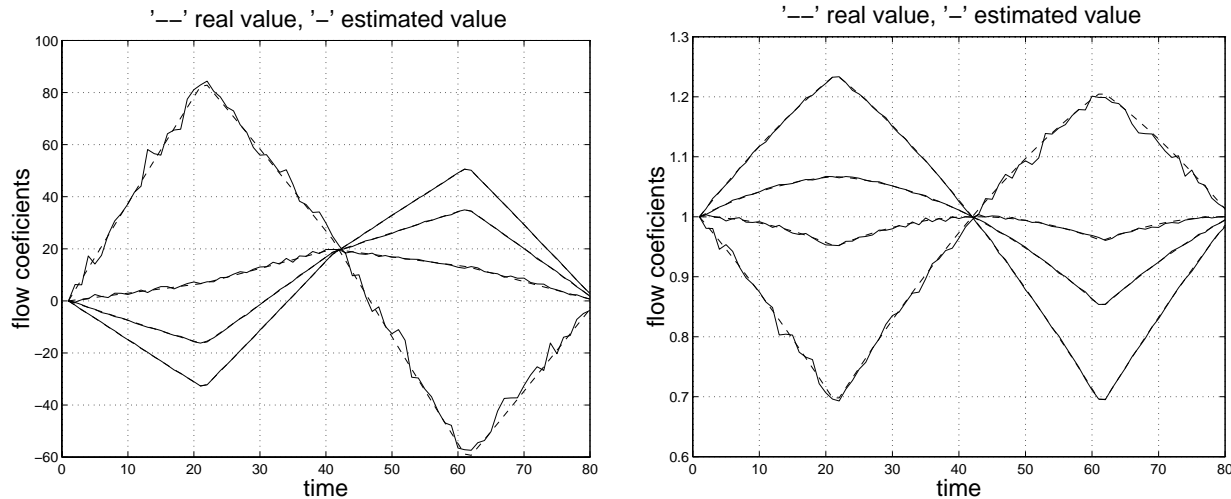


Figure 9.5: Sequence of 1D image motion parameters.

nique to the affine parameters of Figure 9.5. The estimate of the translation along the camera “plane” was obtained by using the 2D version of expression (9.15). The translation estimate is shown on the left plot of Figure 9.4 superimposed with the true translation value. The 2D rotation and the 2D shape were estimated by factorizing the 2D version of the surface-based measurement matrix. The rotation estimate is on the right plot of Figure 9.4 superimposed to the true value of the rotation angle. The 2D shape estimate is represented in Figure 9.3 superimposed to the true 2D shape. The good agreement between the estimates of the 2D structure and the actual 2D structure, see Figures 9.3 and 9.4, illustrate the good performance of the surface-based factorization method.

Weighted factorization

To illustrate the effect of taking into account the 2D motion estimation errors when recovering 3D structure from 2D motion, we synthesized two subsets of trajectories, represented in Figure 9.6, each with a different level of observation noise, and applied both the non-weighted factorization and the weighted factorization methods.

We generated two subsets of, respectively, 10 and 11 feature points with coordinates x and y randomly located inside a square. To facilitate the visualization of the experimental

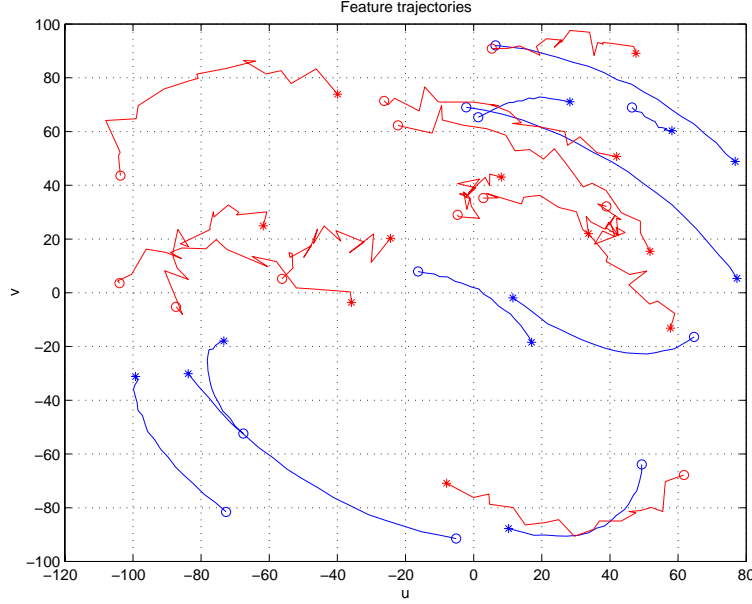


Figure 9.6: Feature trajectories with two levels of observation noise.

results, the depth z was generated with a sinusoidal shape applied to the ordered set of 21 features. The coordinate z is represented in both plots of Figure 9.7 with small circles. The dots on the plots of Figure 9.7 represent estimates of the relative depth. The 3D rotational motion was simulated by synthesizing a smooth time evolution for the Euler angles that specify the orientation of the object coordinate system relative to the camera coordinate system. The time evolution of the 6 entries of the 3D rotation matrix that are involved in the orthogonal projection is represented in both plots of Figure 9.9 with thick lines. The 3D translation is also smooth. The two components of the translation along the camera plane are represented in the plots of Figure 9.8 with thick lines. The thin lines in the plots of Figures 9.9 and 9.8 represent estimates of the 3D motion.

We used the perspective projection model to project the features onto the image plane. The lens focal length parameter was set to a value high enough such that the orthographic projection can be considered a valid approximation. Figure 9.6 shows the feature trajectories on the image plane, after adding noise. For each trajectory, the initial position is marked with “o” and the final position is marked with “*”. The noise variance is $\sigma_1^2 = 1$ for the first subset of 10 features and $\sigma_2^2 = 5$ for the second subset

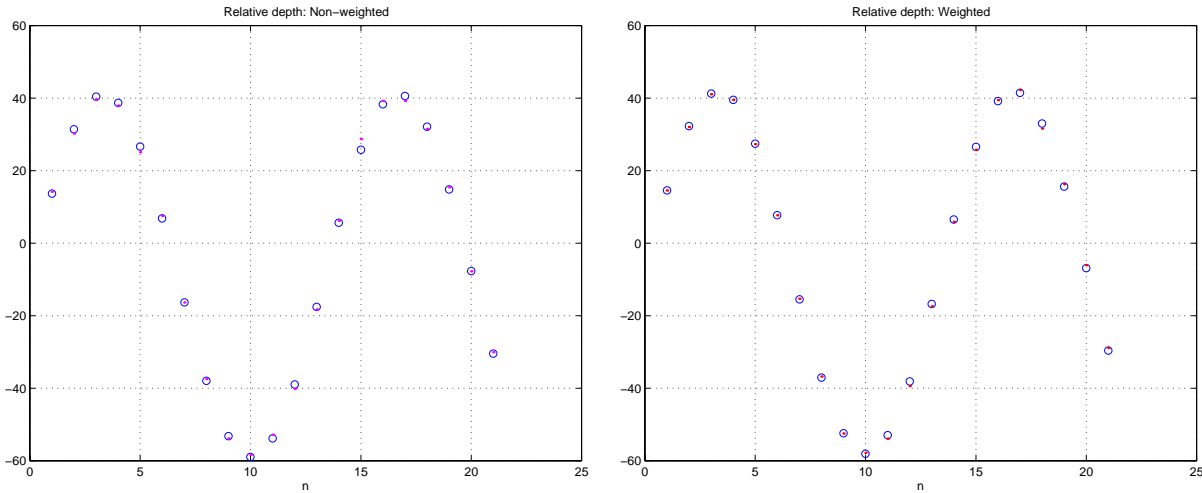


Figure 9.7: Estimates of the relative depth, marked with points, superimposed with the true value of the coordinate z of each feature, marked with small circles. Left plot: estimates obtained with the non-weighted factorization method. Right plot: estimates obtained with the weighted factorization method.

of 11 features. As expected, the trajectories corresponding to the features of the first subset have a smooth evolution, while the trajectories corresponding to the features of the second subset have a more noisy shape, see Figure 9.6.

We applied both the non-weighted feature-based factorization of chapter 8 and the weighted factorization method of section 9.4 to the feature trajectories of Figure 9.6. The estimates of the 3D shape and 3D motion are shown in the plots of Figures 9.7, 9.8, and 9.9, superimposed to the true values. In Figures 9.7, 9.8, and 9.9, the left plots represent the non-weighted estimates, and the right plots represent the weighted factorization results. We see that the 3D motion estimates obtained through the weighted factorization method are more accurate than the ones obtained without taking into account the different noise levels. This is particularly evident for the translation estimates – compare the left and right plots of Figure 9.8. The difference is smaller for the estimates of the entries of the 3D rotation matrix, see Figure 9.9, but these small differences originate much larger differences in the feature projections because the projections are obtained by multiplying the 3D rotation matrix by the 3D position of the features. The 3D shape estimates represented by the relative depths in the left and right plots of Figure 9.7 show a very

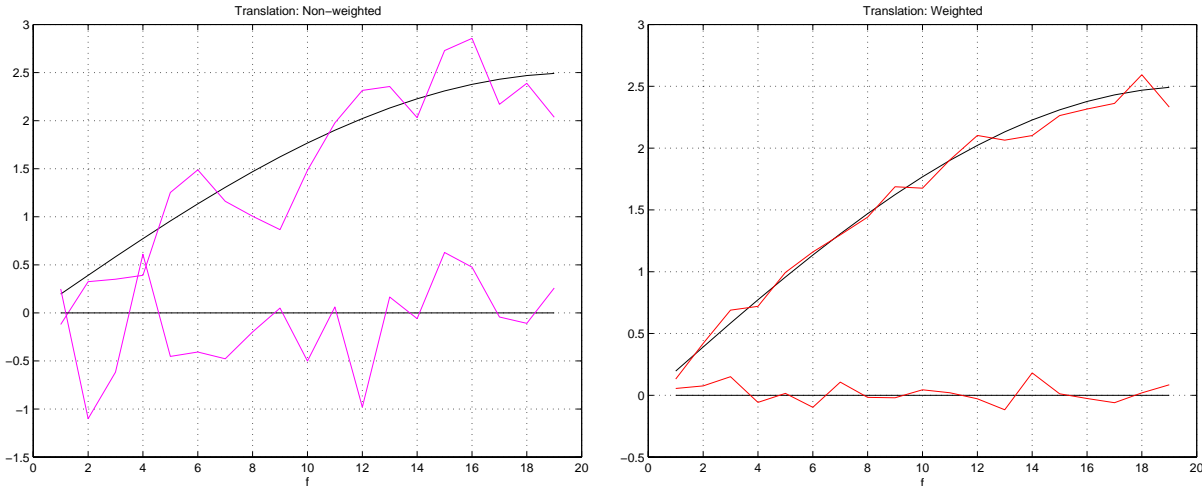


Figure 9.8: Estimates of the translation along the camera plane, marked with thin lines, superimposed to the true translation, marked with thick lines. Left plot: estimates obtained with the non-weighted factorization method. Right plot: estimates obtained with the weighted factorization method.

good agreement with the real 3D shape, both for the non-weighted factorization and the weighted factorization methods. Thus we conclude that, while giving different credit for different trajectories improves the 3D motion estimates, the 3D shape estimate is almost independent of the weights given to the trajectories.

To interpret this behavior, we think of the estimation of the 3D shape as a filtering of the observations across time, and of the estimation of the 3D motion as a filtering of the observations across space. Since the weights given to the observations in the weighted factorization method are different from feature to feature, i.e., less weight is given to the more noisy trajectories, the filtering across space is improved and the weighted estimate of the 3D motion is much more accurate than the non-weighted estimate. In contrast, the weights are constant across time, thus filtering across time is insensitive to the different weights, and the weighted estimate of the 3D shape is similar to the non-weighted estimate.

The filtering analogies in the previous paragraph explain the experimental results in a coarse way. By examining in detail the estimates of the relative depth in both plots of Figure 9.7, we see that the weighted estimate of the 3D shape is slightly more accurate than the non-weighted estimate – compare the very accurate weighted estimates of the

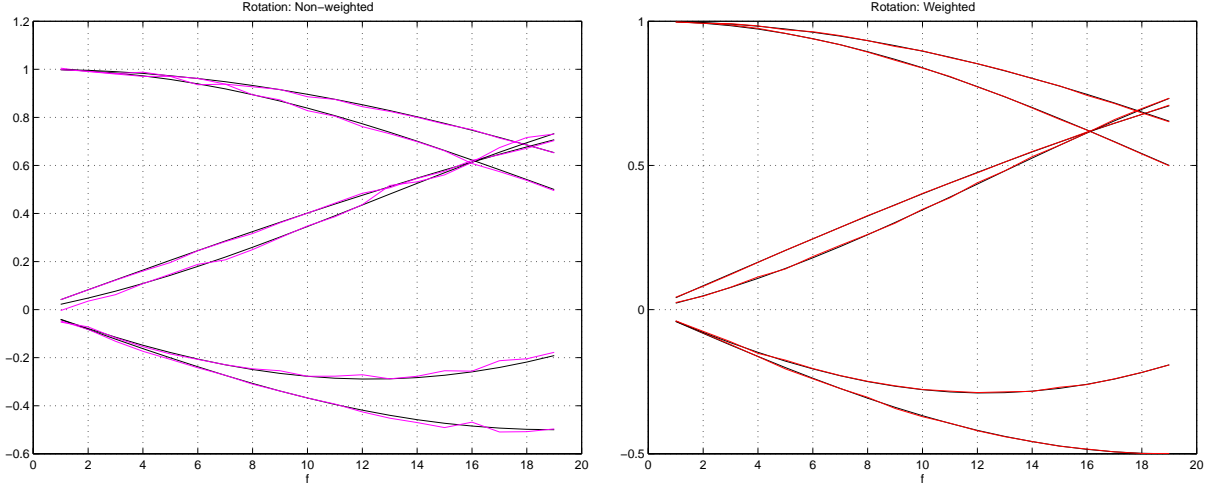


Figure 9.9: Estimates of the 6 entries of the 3D rotation matrix that are involved in the orthogonal projection, marked with thin lines, superimposed to the true values, marked with thick lines. Left plot: estimates obtained with the non-weighted factorization method. Right plot: estimates obtained with the weighted factorization method.

relative depths of the first 10 features (the subset of features observed with lower noise level) in the right plot of Figure 9.7 with the non-weighted estimates in the left plot of Figure 9.7. The accuracy of the weighted estimate of the relative depth of a given feature reflects the level of the observation noise for the trajectory of the projection of that feature – note that the weighted estimates of the relative depths of the subset of features observed with higher level of noise (the last 11 features in the right plot of Figure 9.7) are less accurate than the estimates of the relative depths of the first subset (the first 10 features in the right plot of Figure 9.7).

9.6 Summary

In this chapter we extended the feature-based factorization method introduced in chapter 8. Instead of tracking pointwise features, we track regions where the image motion is described by a single set of parameters. We show how to recover three-dimensional (3D) structure (3D motion and a parametric description of the 3D shape) from the set of parameters describing the two-dimensional (2D) motion of the brightness pattern in the

image plane by using the *surface-based factorization* approach.

In the chapter we also show how to include confidence weights for the parameters describing the 2D motion in the image plane (or for the feature trajectories). We show that the problem is rewritten as the non-weighted factorization of a modified matrix. This way we are able to solve the weighted estimation problem without additional computational cost. We call this extension the *weighted factorization method*.

The experimental results reported in section 9.5 illustrate the good behavior of the algorithms proposed. The first experiment uses an image sequence for which feature tracking is not possible to illustrate the good performance of the surface-based factorization. The second experiment emphasizes the improvement on the accuracy of the estimates of the 3D motion, when the weighted factorization method is used rather than the original non-weighted method.

Chapter 10

Direct Inference of 3D Rigid Shape

10.1 Introduction

In chapter 7 we formulated the problem of inferring three-dimensional (3D) structure (3D motion and 3D shape) from a video sequence by using *Maximum Likelihood* (ML). We showed that the ML cost function depends on the 3D structure through the two-dimensional (2D) motion of the brightness pattern in the image plane. Motivated by this dependence, chapters 8 and 9 dealt with the recovery of 3D structure from the 2D motion induced in the image plane. We used linear subspace constraints to develop efficient factorization methods to recover 3D rigid structure from the 2D motion computed for a set of frames. The factorization approach is well suited to the analysis of scenes that either have texture that makes easy the tracking of feature points or have 3D shape that can be well approximated with polyhedral surfaces.

In this chapter we develop an algorithm that recovers 3D structure directly from the image intensity values. Our algorithm is an approximation to the ML estimation of the unknowns involved in the problem of inferring 3D rigid structure from video. We compute the 3D motion by using the factorization method of chapters 8 and 9. In fact, experiments with real videos show that the 3D rigid motion can be computed with accuracy from the 2D motion computed across a set of frames for a small number of distinguished points or regions. After estimating the 3D motion, we are left with the minimization of the ML cost function with respect to the 3D shape. We propose a computationally

simple *multiresolution continuation-type method* to solve this non-linear minimization. Our algorithm starts by estimating coarse approximations to the 3D shape. Then, it refines the estimate as more images are being taken into account. The computational simplicity of our algorithm comes from the fact that each refinement stage, although non-linear, is solved by a *Gauss-Newton method* that requires no more than two or three iterations. The derivatives involved in the Gauss-Newton iterates are computed from the image gradients in a way similar to the simple common procedure for estimating the 2D motion of the brightness pattern in the image plane.

Our approach provides an efficient way to cope with the ill-posedness of estimating the motion in the image plane. In fact, the local *brightness change constraint* leads to a single restriction, which is insufficient to determine the two components of the local image motion (the so called *aperture problem* discussed in detail in chapter 3). Our method of estimating directly the 3D shape overcomes the aperture problem because we are left with the local depth as a single unknown, after computing the 3D motion in the first step.

Reference [29] also estimates directly the 3D structure parameters by using the brightness change constraint between two consecutive frames. A distinguishing feature of our work is the formulation of the estimate from a set of images, rather than a single pair. This lead to accurate estimates for the 3D structure, due to the 3D rigidity of the scene. Reference [28] builds on the work in reference [29] by using a *Kalman filter* to update the estimates over time. Our approach is distinct from the approach of [28] because we model the rigidity of the scene over a set of frames, instead of trying to fuse a set of possibly inaccurate estimates obtained from pairs of consecutive frames.

Throughout this chapter we assume orthogonal projection to model the image formation. Orthogonal projections have been used as a good approximation to the perspective projection when the object is far from the camera. With this type of scenes, two-frame based methods fail to estimate the absolute depth. Although formulated assuming orthogonal projections, which leads to estimates of the relative depth, our method can be easily extended to cope with perspective projections, which then leads to estimates of the

absolute depth.

This chapter is organized as follows. To motivate the minimization procedure, we first introduce a modified image sequence, obtained from the original sequence and the 3D motion. Section 10.2 introduces this modified image sequence for known motion. In section 10.3, we describe the multiresolution continuation method used to minimize the ML cost function. Illustrative experimental results are in section 10.4. Experiments with real video sequences are described in chapter 11.

10.2 Image Sequence for Known Motion

In chapter 7, we introduced the *Maximum Likelihood* (ML) cost function C_3 involved in the recovery of three-dimensional (3D) rigid structure from video. The cost function C_3 depends on the following unknowns: texture, 3D shape, and 3D motion. In chapter 7, we first maximized it with respect to the unknown texture. After replacing the texture estimate, the ML cost function C_3 is given by expression (7.10), which is written again here for easy reference,

$$C_3(\mathcal{S}, \{\mathbf{m}_f\}) = \sum_{f=2}^F \sum_{g=1}^{f-1} \int \left[\mathbf{I}_f(\mathbf{u}_f(\mathbf{s})) - \mathbf{I}_g(\mathbf{u}_g(\mathbf{s})) \right]^2 \frac{J_f(\mathbf{s})J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})} d\mathbf{s}. \quad (10.1)$$

The ML cost function C_3 in expression (10.1) depends on the 3D shape \mathcal{S} and 3D motion $\{\mathbf{m}_f\}$ only through the two-dimensional (2D) motion induced in the image plane, i.e., through the mappings $\{\mathbf{u}_f(\mathbf{s})\}$. Recall the comment on section 7.2 that $\mathbf{u}_f(\mathcal{S}, \mathbf{m}_f; \mathbf{s})$ depends on the shape \mathcal{S} and the motion \mathbf{m}_f .

Chapters 8 and 9 address the minimization of the functional (10.1) by using the common approach of dividing the overall problem into two steps. The first step estimates the motion in the image plane $\mathbf{u}_f(\mathbf{s})$ by minimizing an approximation of (10.1) (in general, only two frames are taken into account). The second step estimates the shape \mathcal{S} and the motion \mathbf{m}_f from $\{\mathbf{m}_f\}$. Since the motion in the image plane can not be reliably computed in the entire image, these methods cannot provide a reliable dense shape estimate. The approach described in this chapter combines the good performance of the

factorization method in estimating the 3D motion with the robustness of minimizing the ML cost function with respect to the object shape. We recover the 3D motion \mathbf{m}_f as described in chapters 8 and 9. Then, we insert the 3D motion estimates into the ML cost function (10.1), and minimize it with respect to the unknown shape \mathcal{S} .

We start by making explicit the relation between the mapping $\mathbf{u}_f(\mathbf{s})$ that describes the 2D motion in the image plane and the 3D shape \mathcal{S} and the 3D motion \mathbf{m}_f .

Choose the coordinate \mathbf{s} of the generic point in the object surface to coincide with the coordinates $[x, y]^T$ of the object coordinate system. Also, choose the object coordinate system so that it coincides with the camera coordinate system in the first frame. Under orthography, a point with coordinate \mathbf{s} in the object surface is projected on coordinate $\mathbf{u} = [x, y]^T = \mathbf{s}$ in the first frame, so that $\mathbf{u}_1(\mathbf{s}) = \mathbf{s}$. At instant f , that point is projected, according to expressions (9.10,9.11), onto the image coordinate

$$\begin{aligned} \mathbf{u}_f(\mathbf{s}) = \mathbf{u}_f \left(\begin{bmatrix} x \\ y \end{bmatrix} \right) &= \begin{bmatrix} i_{xf} & i_{yf} & i_{zf} \\ j_{xf} & j_{yf} & j_{zf} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_{uf} \\ t_{vf} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{N}_f & \mathbf{n}_f \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ z \end{bmatrix} + \mathbf{t}_f \\ &= \mathbf{N}_f \mathbf{s} + \mathbf{n}_f z + \mathbf{t}_f. \end{aligned} \tag{10.2}$$

Expression (10.2) makes explicit the relation between the mapping $\mathbf{u}_f(\mathbf{s})$ that describes the 2D motion in the image plane and the 3D motion, represented by $\{\mathbf{N}_f, \mathbf{n}_f\}$, and the 3D shape, represented by the unknown relative depth z .

The 3D shape and the 3D motion are observed in a coupled way through the 2D motion on the image plane as expression (10.2) shows. When the 3D motion is known, the problem of inferring the 3D shape from the image sequence is simplified. In fact, the local brightness change constraint leads to a single restriction, which is insufficient to determine the two components of the local image motion (the so called aperture problem studied in detail in chapter 3). Our method of estimating directly the 3D shape overcomes the aperture problem because we are left with the local depth as a single unknown, after computing the 3D motion in the first step. To better illustrate why the problem

becomes much simpler when the 3D motion is known, we introduce a modified image sequence $\{\tilde{\mathbf{I}}_f, 1 \leq f \leq F\}$, obtained from the original sequence $\{\mathbf{I}_f, 1 \leq f \leq F\}$ and the 3D motion. We show that the 2D motion of the brightness pattern on image sequence $\tilde{\mathbf{I}}_f$ depends on the 3D shape in a very particular way. This motivates the algorithm we use to minimize the ML cost function in expression (10.1).

Consider the image $\tilde{\mathbf{I}}_f$ related to image \mathbf{I}_f by the following affine mapping that depends only on the 3D position at instant f ,

$$\tilde{\mathbf{I}}_f(\mathbf{s}) = \mathbf{I}_f(\mathbf{N}_f \mathbf{s} + \mathbf{t}_f). \quad (10.3)$$

From this definition it follows that a point \mathbf{s} , which projects to coordinate $\mathbf{u}_f(\mathbf{s})$ in image \mathbf{I}_f , is mapped to coordinate

$$\tilde{\mathbf{u}}_f(\mathbf{s}) = \mathbf{N}_f^{-1} [\mathbf{u}_f(\mathbf{s}) - \mathbf{t}_f] \quad (10.4)$$

in image $\tilde{\mathbf{I}}_f$. Replacing $\mathbf{u}_f(\mathbf{s})$ by expression (10.2), we obtain for the 2D motion of the brightness pattern of the modified sequence $\{\tilde{\mathbf{I}}_f\}$,

$$\tilde{\mathbf{u}}_f(\mathbf{s}) = \mathbf{s} + \mathbf{N}_f^{-1} \mathbf{n}_f z. \quad (10.5)$$

Expression (10.5) shows that the trajectory of a point \mathbf{s} in image sequence $\{\tilde{\mathbf{I}}_f\}$ depends on the relative depth of that point in a very particular way. In fact, the trajectory has the same shape for every point. The shape of the trajectories is given by the evolution of the 2×1 vector $\mathbf{N}_f^{-1} \mathbf{n}_f$ across the frame index f . Thus, the shape of the trajectories depends uniquely on the rotational component of the 3D motion. The relative depth z affects only the magnitude of the trajectory. A point with relative depth $z = 0$ is stationary in image sequence $\{\tilde{\mathbf{I}}_f\}$, since we get $\tilde{\mathbf{u}}_f(\mathbf{s}) = \mathbf{s}$ for arbitrary 3D motion of the object, by making $z = 0$ in expression (10.5). Clearly, this is not the case in the original image sequence $\{\mathbf{I}_f\}$, as shown by expression (10.2).

10.3 Minimization Procedure: Continuation Method

By minimizing the *Maximum Likelihood* (ML) cost function in expression (10.1) with respect to the relative depth z of each point \mathbf{s} , we are in fact estimating the magnitude of the trajectory of the point to where the point \mathbf{s} maps in image sequence $\{\tilde{\mathbf{I}}_f\}$. The shape of the trajectory is known, since it depends only on the three-dimensional (3D) motion.

Our algorithm is based on this characteristic of the ML cost function. We use a *multiresolution continuation-type method* to estimate the relative depth of each point. The algorithm refines the estimate of the relative depth as more frames are being taken into account. When just a few frames are taken into account, the magnitude of the trajectories in the image sequence $\{\tilde{\mathbf{I}}_f\}$ can be only roughly estimated because the length of the trajectories is short and their shape may be quite simple. When enough frames are considered, the trajectories on image sequence $\{\tilde{\mathbf{I}}_f\}$ are long enough, their magnitude is unambiguous, and the relative depth estimates are accurate. Our algorithm does not compute the image sequence $\{\tilde{\mathbf{I}}_f\}$, it rather uses the corresponding intensity values of the original sequence $\{\mathbf{I}_f\}$.

The advantage of the continuation-type method is that it provides a computationally simple way to estimate the relative depth because each stage of the algorithm updates the estimate by using a *Gauss-Newton method*, i.e., by solving a linear problem. We consider the relative depth z to be constant in a region \mathcal{R} . We estimate z by minimizing the energy resultant from neglecting the weighting factor $\frac{J_f(\mathbf{s})J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})}$ in the ML cost function (10.1).

We define $e(z; \mathbf{s})$ as the integrand of the ML cost function (10.1) after neglecting the weighting factor $\frac{J_f(\mathbf{s})J_g(\mathbf{s})}{\sum_{h=1}^F J_h(\mathbf{s})}$,

$$e(z; \mathbf{s}) = \mathbf{I}_f(\mathbf{N}_f \mathbf{s} + \mathbf{n}_f z + \mathbf{t}_f) - \mathbf{I}_g(\mathbf{N}_g \mathbf{s} + \mathbf{n}_g z + \mathbf{t}_g). \quad (10.6)$$

The estimate \hat{z} of the relative depth z for the region \mathcal{R} is then

$$\hat{z} = \arg \min_z E(z), \quad (10.7)$$

where $E(z)$ is the energy function

$$E(z) = \sum_{f=2}^F \sum_{g=1}^{f-1} \int_{\mathcal{R}} e^2(z; \mathbf{s}) d\mathbf{s}. \quad (10.8)$$

We compute the estimate \hat{z} by refining a previous estimate z_0 , as

$$\hat{z} = z_0 + \hat{\delta}_z, \quad \text{with} \quad \hat{\delta}_z = \arg \min_{\delta_z} E(z_0 + \delta_z). \quad (10.9)$$

The Gauss-Newton method neglects the second and higher order terms of the Taylor series expansion of $e(z_0 + \delta_z; \mathbf{s})$ for fixed \mathbf{s} ,

$$e(z_0 + \delta_z; \mathbf{s}) \simeq e(z_0; \mathbf{s}) + e'(z_0; \mathbf{s})\delta_z, \quad (10.10)$$

where $e'(z_0; \mathbf{s})$ is the partial derivative of $e(z; \mathbf{s})$ with respect to z , evaluated at $z = z_0$.

By making this approximation, we get the following expression for the increment $\hat{\delta}_z$ that minimizes (10.9),

$$\hat{\delta}_z = - \frac{\sum_{f=2}^F \sum_{g=1}^{f-1} \int_{\mathcal{R}} e(z_0; \mathbf{s}) e'(z_0; \mathbf{s}) d\mathbf{s}}{\sum_{f=2}^F \sum_{g=1}^{f-1} \int_{\mathcal{R}} [e'(z_0; \mathbf{s})]^2 d\mathbf{s}}, \quad (10.11)$$

The derivative $e'(z; \mathbf{s})$ is computed from the spatial gradient of the images in a way similar to the simple common procedure for estimating the two-dimensional (2D) motion of the brightness pattern in the image plane. By differentiating $e(z; \mathbf{s})$, given by expression (10.6), with respect to z , we express $e'(z; \mathbf{s})$ in terms of the components of the spatial gradient of images \mathbf{I}_f and \mathbf{I}_g as

$$\begin{aligned} e'(z; \mathbf{s}) = & \mathbf{I}_{f_x}(\mathbf{N}_f \mathbf{s} + \mathbf{n}_f z + \mathbf{t}_f) i_{zf} + \mathbf{I}_{f_y}(\mathbf{N}_f \mathbf{s} + \mathbf{n}_f z + \mathbf{t}_f) j_{zf} \\ & - \mathbf{I}_{g_x}(\mathbf{N}_g \mathbf{s} + \mathbf{n}_g z + \mathbf{t}_g) i_{zg} - \mathbf{I}_{g_y}(\mathbf{N}_g \mathbf{s} + \mathbf{n}_g z + \mathbf{t}_g) j_{zg}, \end{aligned} \quad (10.12)$$

where \mathbf{I}_{f_x} and \mathbf{I}_{f_y} denote the components of the spatial gradient of image \mathbf{I}_f , and i_z , and j_z are the entries of the rotation matrix introduced in chapter 8, section 8.2, expression (8.1).

At the beginning, we start with the initial guess $z_0 = 0$ for any region \mathcal{R} . We use square regions where z is estimated as being constant. The size of the regions determines the resolution of the depth estimate. We use large regions when processing the first frames and decrease the size of the regions as the continuation method takes more frames into account.

10.4 Experiment

We describe one experiment that illustrates our approach. This experiment uses a synthetic sequence for which we compare the estimates obtained with the ground truth. In chapter 11 we use the multiresolution continuation algorithm to recover 3D shape from a real life video clip. In this section, we consider that the world is two-dimensional (2D) and that the images are one-dimensional (1D) orthogonal projections of the world. This scenario was introduced in chapter 9 and illustrated in Figure 9.1. It reflects all the basic properties and difficulties of the problem of recovering rigid structure from video and corresponds to the real three-dimensional (3D) world if we consider only one epipolar plane and assume that the motion occurs on that plane.

From expression (9.2), we see that the point (x, z) projects at time f on the image coordinate u_f given by

$$u_f = i_{xf}x + i_{zf}z + t_{uf} \quad (10.13)$$

Expression (10.13) shows that the orthogonal projection is insensitive to the translation component t_{wf} of the object motion.

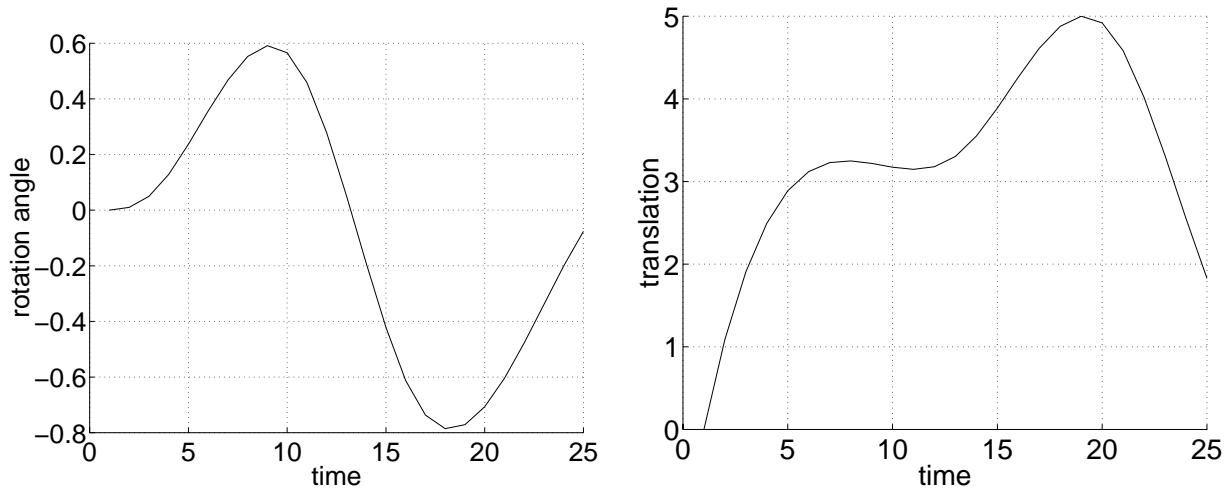


Figure 10.1: True motion. Left: rotational motion; right: translational motion.

Using the constructs represented in Figures 10.1 and 10.2, we generated the image sequence of Figure 10.3. The time evolution of the translational and rotational components

of the motion are shown respectively on the left and right plots of Figure 10.1. The object shape is shown on the plot of Figure 10.2. The object texture is an intensity function defined over the object contour.

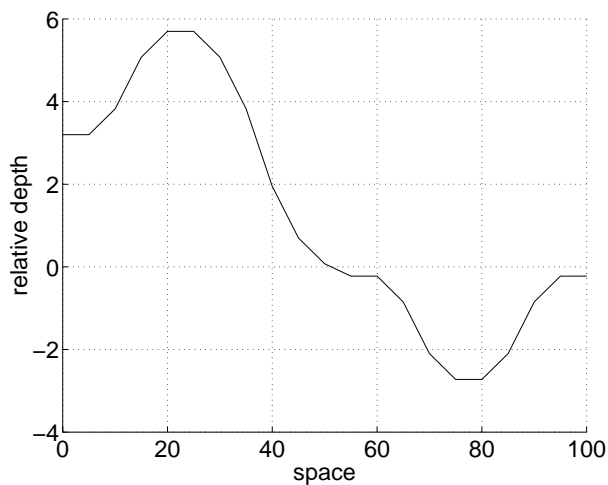


Figure 10.2: True shape.

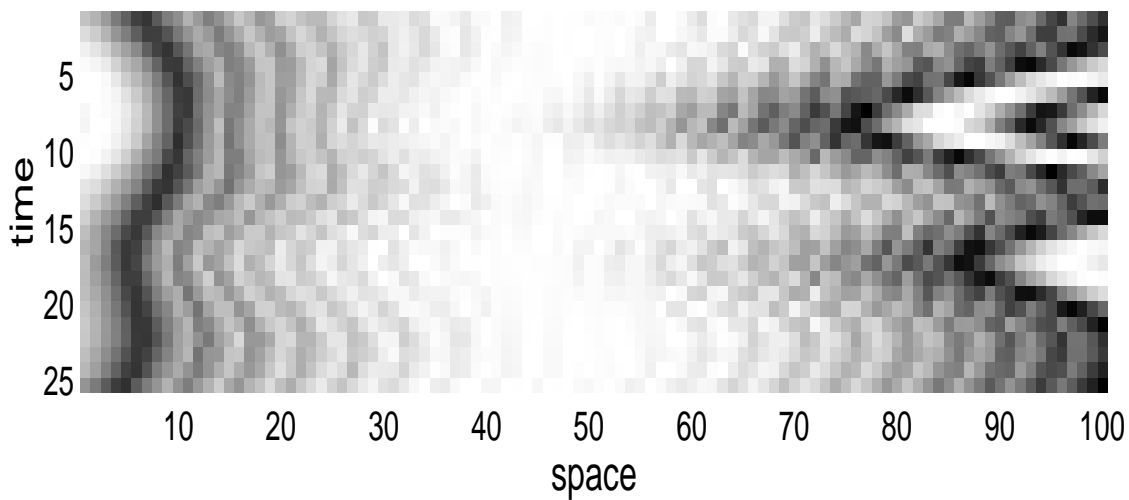


Figure 10.3: Sequence of 25 1D images: each horizontal slice is one image.

In Figures 10.3 and 10.4 we show the computer generated sequence of 25 1D images. Time increases from top to bottom. We obtained the image sequence in Figure 10.3 by projecting the object texture on the image plane according to expression (10.13) and by

adding noise. The surface plotted in Figure 10.4 represents the brightness intensity value as a function of the pixel index and the time instant.

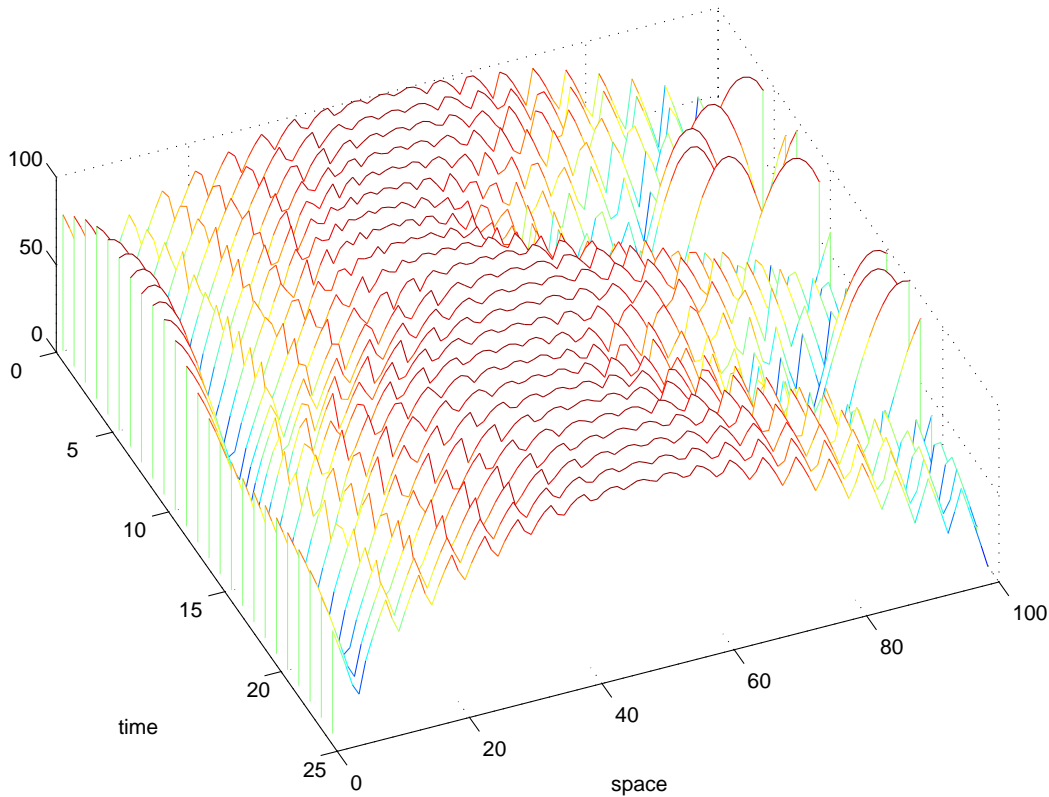


Figure 10.4: Sequence of 25 1D images: evolution of the brightness intensity value.

In Figure 10.5 we represent the modified image sequence, computed from the original sequence in Figures 10.3, as described in section 10.2 for the 3D scenario, see expression (10.3). The motion of the brightness pattern in Figure 10.5 is simpler than the motion in Figure 10.3. In fact, the horizontal positions of the brightness patterns in Figure 10.5 have a time evolution that is equal for the entire image (see, from Figure 10.5 and the plot of Figure 10.2, that the shape of the trajectories of the brightness patterns is related to the rotational component of the motion). Only the amplitude of the time evolution of the horizontal positions of the brightness patterns in Figure 10.5 is different from one object region to another object region. The amplitude for a given region is proportional to the relative depth of that region. Note that the brightness pattern is almost

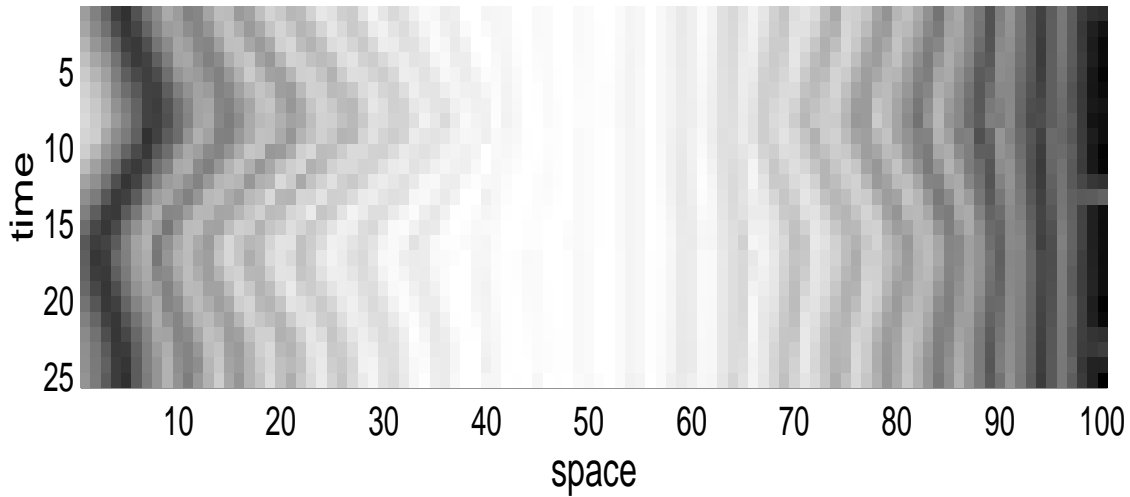


Figure 10.5: Image sequence for known motion.

stationary for regions with relative depth close to zero (see the regions around pixels 55 and 95 on the plot of Figure 10.2 and on Figure 10.5). This agrees with the discussion in section 10.2. In Figure 10.6 we represent the evolution of the brightness intensity value of the modified image sequence as a function of the pixel index and the time instant. The above mentioned differences between the time evolution of the brightness pattern of the modified image sequence and the one of the original sequence are also evident by comparing the surface represented in Figure 10.6 with the brightness surface for the original image sequence, represented in Figure 10.4.

We estimated the relative depth of the object by using the *multiresolution continuation-type method* introduced in section 10.3. The evolution of the relative depth estimate is represented in the plots of Figure 10.7 for several time instants. The size of the estimation region \mathcal{R} was 10 pixels when processing the first 5 frames, 5 pixels when processing frames 6 to 10, and 3 pixels when processing frames 11 to 25. The top left plot was obtained with the first three frames and shows a very coarse estimate of the shape. The bottom right plot was obtained after all 25 frames of the image sequence have been processed. In this plot we made a linear interpolation between the central points of consecutive estimation regions. This plot superimposes the true and the estimated depths

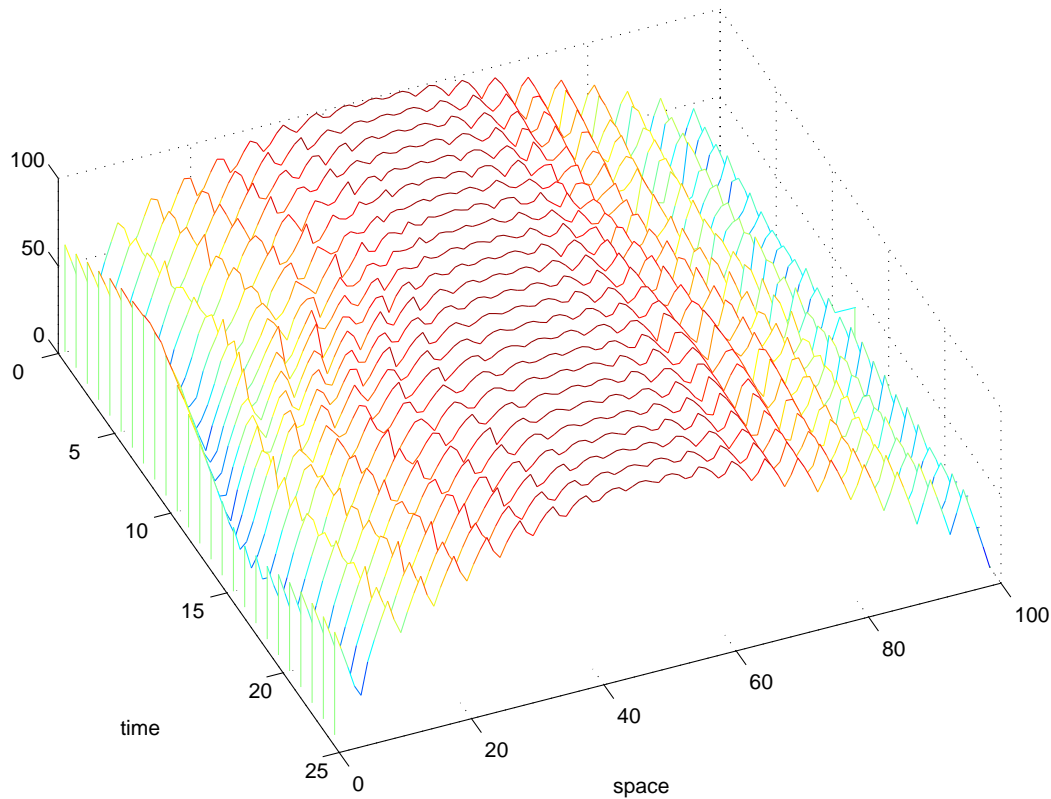


Figure 10.6: Image sequence for known motion: evolution of the brightness intensity value.

showing a very good agreement between them. The intermediate plots show progressively better estimates of the depth shape.

10.5 Summary

In this chapter we develop an algorithm that recovers three-dimensional (3D) rigid shape directly from the image intensity values of a two-dimensional (2D) video sequence.

The problem was formulated as the *Maximum Likelihood* (ML) estimation of the unknown texture, 3D shape, and 3D motion. After replacing for the ML estimate of the texture, we estimate the 3D motion by using the factorization methods of chapters 7 and 8. In this chapter we develop a *multiresolution continuation-type* algorithm to minimize the ML cost function with respect to the object 3D shape. The algorithm starts by estimating coarse approximations to the 3D shape. Then, it refines the estimate as more

images are being taken into account.

Our approach is computationally simple because each stage of the continuation-type algorithm is solved by using a *Gauss-Newton method* that requires no more than two or three iterations. The derivatives involved in the Gauss-Newton iterates are computed from the image gradients in a way similar to the simple common procedure for estimating the 2D motion of the brightness pattern in the image plane.

The experimental results described in section 10.4 illustrate that the algorithm succeeds in recovering dense estimates of the rigid shape.

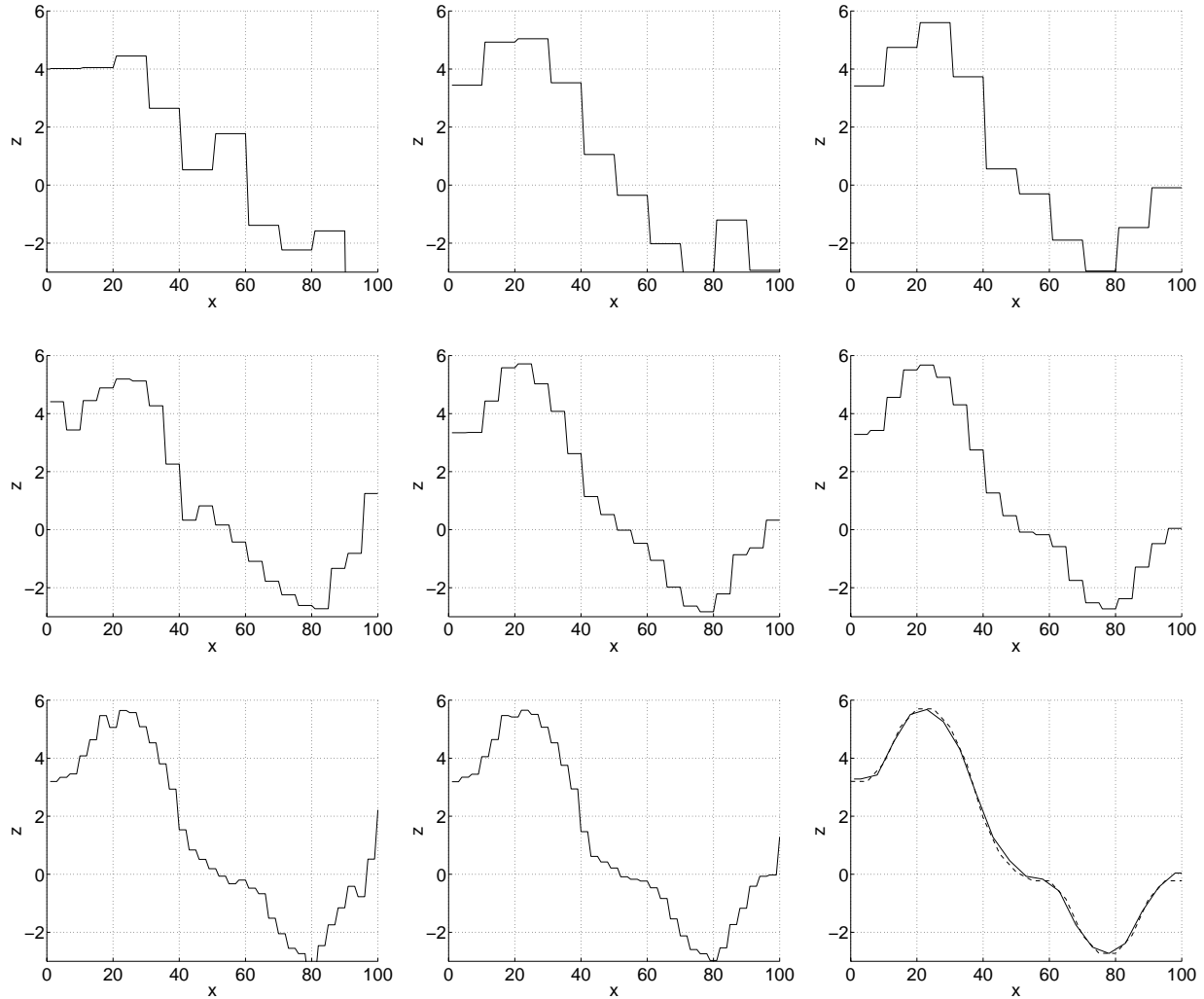


Figure 10.7: Continuation method: evolution of the shape estimate for the image sequence represented in Figure 10.3. Left to right, top to bottom, after processing N frames where N is successively: 3, 4, 5, 6, 8, 10, 15, 20, 25. The bottom right plot superimposes the true shape (dashed line) and its estimate (solid line).

Chapter 11

Real Video Experiments

In this chapter we describe three experiments that recover three-dimensional (3D) structure from real life video sequences. The first two experiments, in sections 11.1 and 11.2, demonstrate the performance of the *surface-based rank 1 factorization method* described in chapters 8 and 9. In the last experiment, described in section 11.3, we use the *multiresolution continuation-type method* developed in chapter 10.

11.1 Box

In this experiment, we used a hand held taped video sequence of 30 frames showing a box over a carpet. Figure 11.1 shows three consecutive frames of the box video sequence. The 3D shape of the scene is well described in terms of four planar patches. One corresponds to the floor, and the other three correspond to the three visible faces of the box. The camera motion was approximately a rotation around the box.



Figure 11.1: Three consecutive frames of the box video sequence.

We processed the box video sequence by using the *surface-based rank 1 factorization method*. We start by estimating the parameters describing the 2D motion of the brightness pattern in the image plane. As derived in chapter 9, for planar patches, the 2D motion

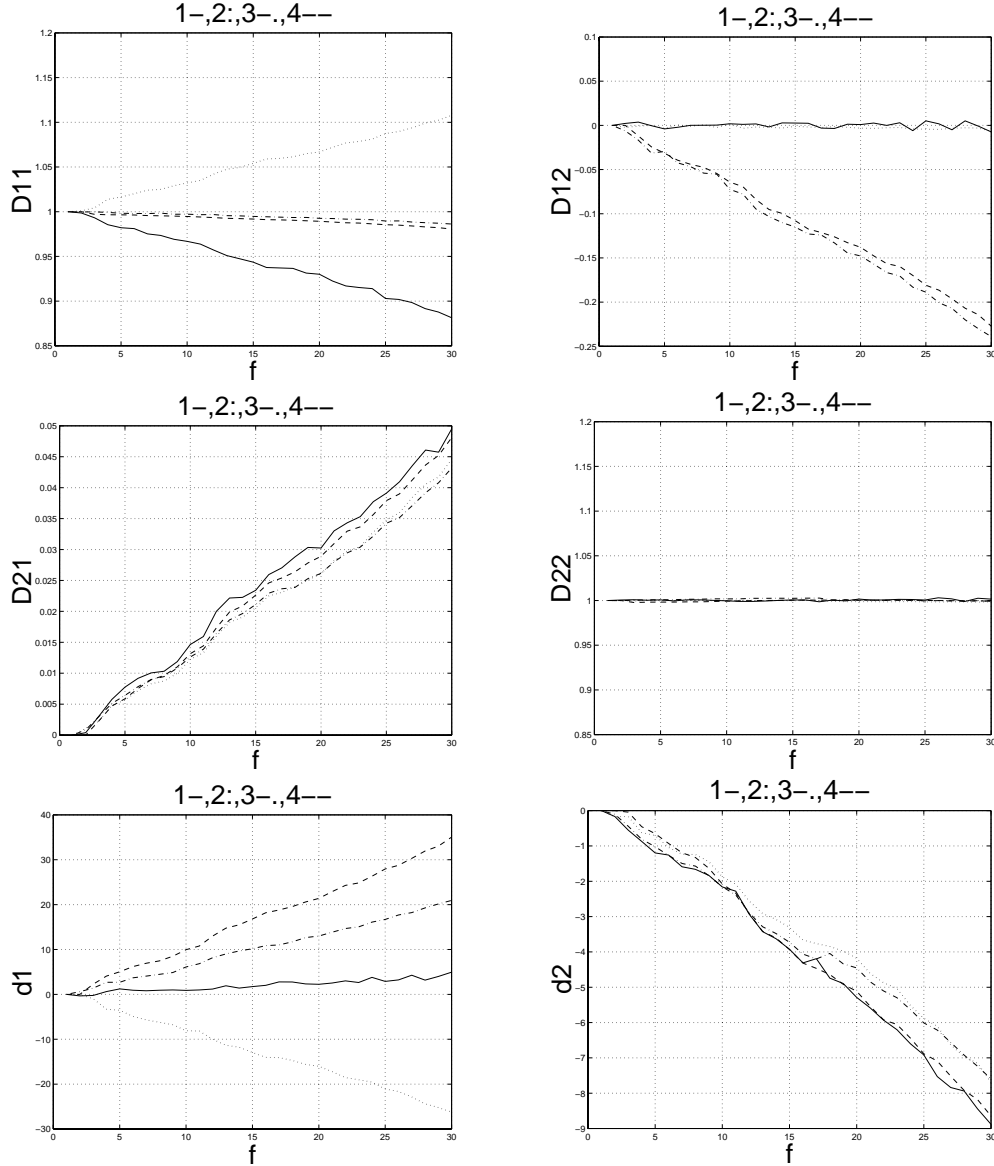


Figure 11.2: Estimates of the image motion parameters in the 2×2 matrix \mathbf{D}_f^n and the 2×1 vector \mathbf{d}_f^n . From left to right, top to bottom, \mathbf{D}_{11} , \mathbf{D}_{12} , \mathbf{D}_{21} , \mathbf{D}_{22} , \mathbf{d}_1 , and \mathbf{d}_2 .

in the image plane is described by the affine motion model. To segment the regions corresponding to different planar patches, we used the method outlined in chapter 3, i.e., we slid a 20×20 window across the image and detected abrupt changes in the affine

motion parameters. We estimated the affine motion parameters as detailed in chapter 3.

The plots in Figure 11.2 represent the time evolution of the affine motion parameters. The 6 affine motion parameters are the entries of the 2×2 matrix \mathbf{D}_f^n and the 2×1 vector \mathbf{d}_f^n introduced in chapter 9, section 9.3, see expression (9.14). The top four plots of Figure 11.2 represent the entries of \mathbf{D}_f^n as a function of f for each of the four planar patches. The bottom two plots represent \mathbf{d}_f^n . We used four different line types to identify each of the planar patches. The solid line corresponds to patch 1 (the left side vertical face of the box in the frames of Figure 11.1). The dotted line corresponds to patch 2 (the right side vertical face of the box). The dash-dotted line corresponds to patch 3 (the top of the box). The dashed line corresponds to patch 4 (the floor). We see the evolution of the set of affine parameters is distinct for each surface patch, in particular see the evolution of \mathbf{D}_{11} , \mathbf{D}_{12} , and \mathbf{d}_1 .

From the affine motion parameters of Figure 11.2, we have recovered the 3D structure of the scene by using the *surface-based rank 1 factorization method* described in chapters 8 and 9. For the box video sequence, the shape matrix \mathbf{S} contains four submatrices, one for each planar patch,

$$\mathbf{S}^T = \begin{bmatrix} \mathbf{S}_1^T & \mathbf{S}_2^T & \mathbf{S}_3^T & \mathbf{S}_4^T \end{bmatrix}. \quad (11.1)$$

Each submatrix in expression (11.1) is a matrix that contains the 3D shape parameters of the corresponding surface patch. The general structure of matrix \mathbf{S}_n was specified in chapter 9, section 9.3, see expression (9.17). In this experiment, expression (9.17) is particularized to planar surface patches, leading to the 3×3 matrix

$$\mathbf{S}_n^T = \begin{bmatrix} x_0^n & 1 & 0 \\ y_0^n & 0 & 1 \\ a_{00}^n & a_{10}^n & a_{01}^n \end{bmatrix}, \quad (11.2)$$

where (x_0^n, y_0^n) are the coordinates of the centroid of the support region of patch n and $\{a_{00}^n, a_{10}^n, a_{01}^n\}$ are the parameters describing the 3D shape of the patch by $z = a_{00}^n + a_{10}^n(x - x_0^n) + a_{01}^n(y - y_0^n)$.

We computed the parameters describing the 3D structure, i.e, the 3D motion parameters $\{t_{uf}, t_{vf}, \theta_f, \phi_f, \psi_f, 1 \leq f \leq 30\}$ and the 3D shape parameters $\{a_{00}^n, a_{10}^n, a_{01}^n, 1 \leq n \leq 4\}$

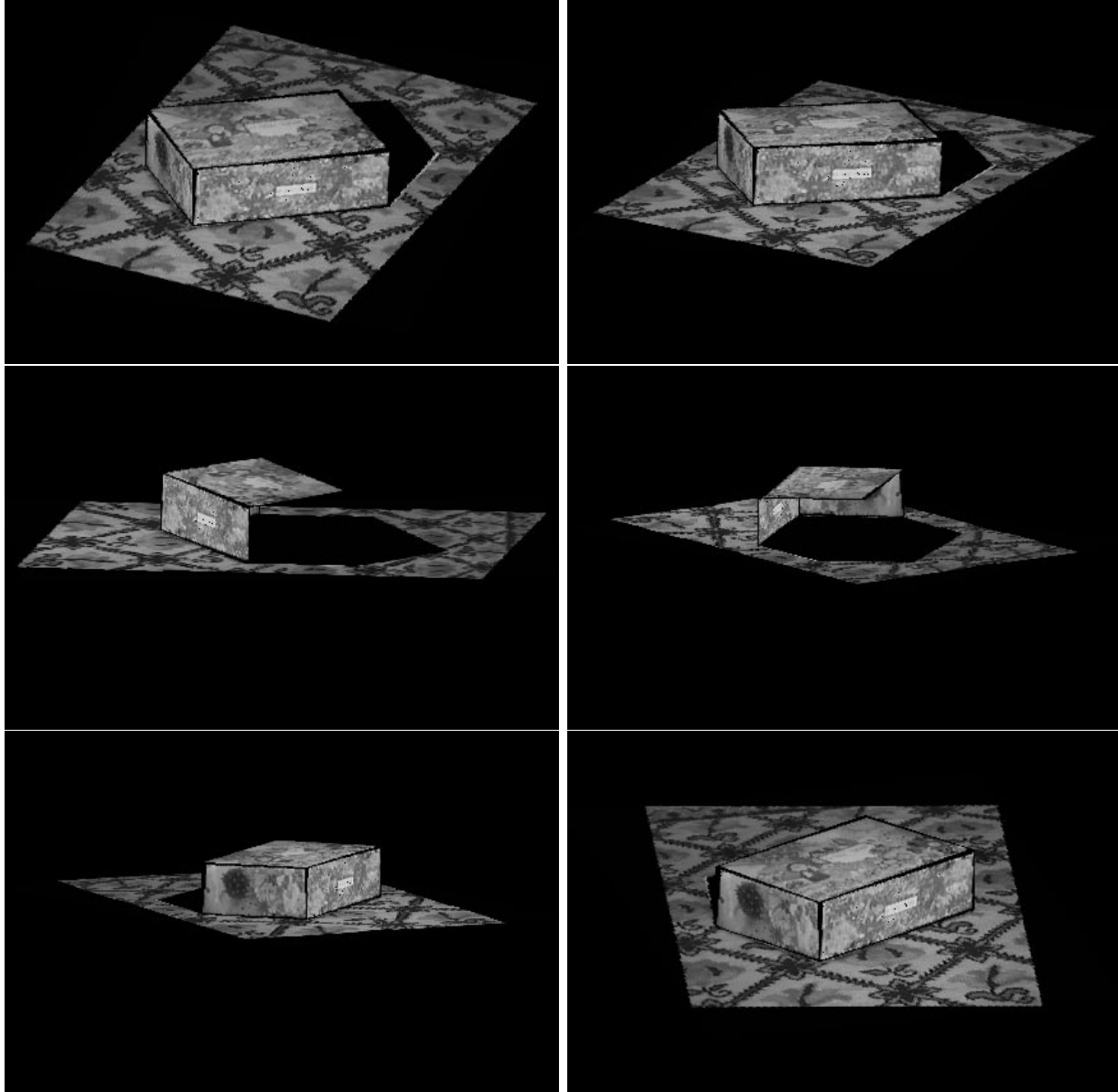


Figure 11.3: Six perspective views of the 3D shape and texture reconstructed from the box video sequence of Figure 11.1.

from the image motion parameters in $\{\mathbf{D}_f^n, \mathbf{d}_f^n, 1 \leq f \leq 30, 1 \leq n \leq 4\}$ by using the surface-based rank 1 factorization method.

After computing the 3D structure parameters, we recover the texture of each surface patch by averaging the video frames co-registered according to the recovered 3D structure.

Figure 11.3 shows six perspective views of the reconstructed 3D shape with the scene texture mapped on it. The spatial limits of the planar patches were determined the following way. Each edge that links two visible patches was computed from the intersection of the planes corresponding to the patches. Each edge that is not in the intersection of two visible patches was computed by fitting a line to the boundary that separates two regions with different 2D motion parameters. We see that the angles between the planar patches are correctly recovered.

Video compression example

Model-based video representations enable very low bit rate compression. Basically, instead of representing a video sequence in terms of frames and pixels, 3D model-based approaches use the recovered 3D structure. A video sequence is then represented by the 3D shape and texture of the object, and its 3D motion. Within the surface-based representation, the 3D motion and 3D shape are coded with a few parameters and the texture is coded as a set of ordinary images, one for each planar patch. We use the box video sequence to illustrate this video compression scheme.

The video analysis task consists in recovering the object shape, object motion, and object texture from the given video. The steps of the analysis task for the box video sequence were detailed above. Figure 11.4 shows frontal views of the four elemental texture constructs of the surface-based representation of the box video sequence. On the left, the planar patch corresponding to the carpet is not complete. This is because the region of the carpet that is occluded by the box can not be recovered from the video sequence. The other three images in Figure 11.4 are the three faces of the box.

The video synthesis task consists in generating a video sequence from the recovered



Figure 11.4: The four planar patches that constitute the elemental texture constructs. From the left to the right, the carpet (floor level), and the three visible faces of the box: top of the box, the right side of the box, and the left side of the box.

3D motion, 3D shape, and texture of the object. The synthesis task is much simpler than the analysis because it involves only an appropriate warping of the recovered object texture. The frame \mathbf{I}_f is synthesized by projecting the object texture according to the model introduced in chapter 7,

$$\mathbf{I}_f = \mathcal{P}\left\{\mathcal{M}(\mathbf{m}_f)\mathcal{O}\right\} = \mathcal{T}(\mathbf{s}_f(\mathbf{u})). \quad (11.3)$$

The projection of the texture is straight forward because it involves only the warping of the texture image according to the mapping $\mathbf{s}_f(\mathbf{u})$, the inverse of $\mathbf{u}_f(\mathbf{s})$. The operations that must be carried out to synthesize the region corresponding to surface patch n at time instant f are: from the 3D shape parameters $\{a_{00}^n, a_{10}^n, a_{01}^n\}$ and the 3D motion parameters $\{t_{uf}, t_{vf}, \theta_f, \phi_f, \psi_f\}$, compute the parameters $\mathbf{D}_f^n, \mathbf{d}_f^n$ describing the affine mapping $\mathbf{u}_f(\mathbf{s})$, see chapter 9, section 9.3, expressions (9.13) and (9.14); then, project the texture of the patch k according to the inverse affine mapping $\mathbf{s}_f(\mathbf{u})$,

$$\mathbf{s}_f(\mathbf{u}) = \mathbf{s}(\mathbf{D}_f^{n\#}, \mathbf{d}_f^{n\#}; \mathbf{u}) = \mathbf{D}_f^{n\#} \mathbf{u} + \mathbf{d}_f^{n\#}, \quad (11.4)$$

$$\text{where } \mathbf{D}_f^{n\#} = (\mathbf{D}_f^n)^{-1} \quad \text{and} \quad \mathbf{d}_f^{n\#} = -(\mathbf{D}_f^n)^{-1} \mathbf{d}_f^n + \mathbf{s}_0^n. \quad (11.5)$$

The original sequence has $50 \times 320 \times 240 = 3840000$ bytes. The representation based on the 3D model needs $\sum_n T_n + \sum_n S_n + 50 \times M = \sum_n T_n + 2248$ bytes, where T_n is the storage size of the texture of patch n , S_n is the storage size of the shape of patch n , and M is the storage size of each camera position. Since the temporal redundancy was eliminated, the compression ratio chosen for the spatial conversion governs the overall video compression ratio. To compress the texture of each surface patch in Figure 11.4, we used the JPEG standard with two different compression ratios. The storage sizes T_1 , T_2 , T_3 , and T_4 of the texture patches were, in bytes, from left to right in Figure 11.4, 2606, 655, 662, and 502, for the higher compression ratio and 6178, 1407, 1406, and 865, for the lower compression ratio. These storage sizes lead to the average spatial compression ratios of 31:1 and 14:1.

The first frame of the original box video sequence is on the left side of Figure 11.5. The center and right images show the first frame of the synthesized sequence for the two different JPEG spatial compression ratios. In the center image, the first frame obtained with the higher spatial compression ratio, leading to the overall video compression ratio of 575:1. The right image corresponds to the lower spatial compression ratio and an overall video compression ratio of 317:1. In both cases, the compression ratio due to the elimination of the temporal redundancy is approximately 20.

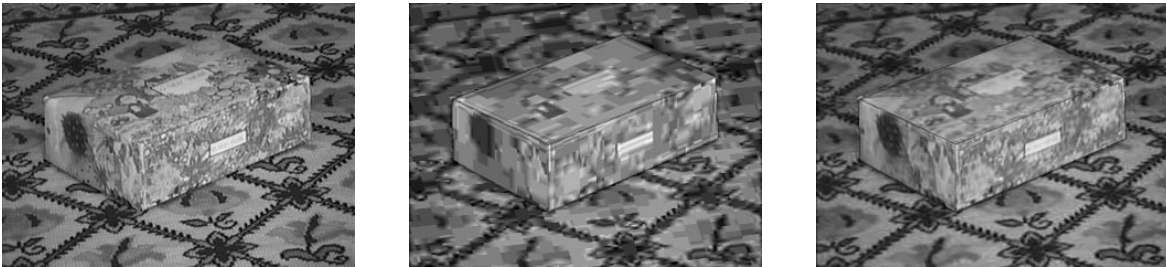


Figure 11.5: Video compression. Left: frame 1 of the box video sequence, Center: frame 1 of the synthesized sequence for a compression ratio of 575:1, Center: frame 1 of the synthesized sequence coded for a compression ratio of 317:1.

From the compressed frames in Figure 11.5 we see that the overall quality is good but there are small artifacts in the boundaries of the surface patches.

11.2 Pedestal

In this experiment, we used a video showing a pedestal. Figure 11.6 shows two frames from a sequence of 25 images. The shape of the part of the pedestal that is captured in the video is piecewise planar with nine planar patches.



Figure 11.6: Two frames from the pedestal video sequence.

After hand-segmenting the region corresponding to the pedestal, we applied the *surface-based rank 1 factorization method*. First, we computed the nine sets of parameters of the 2D affine motion models that describe the motion of the brightness pattern in the image plane. Then, we used the surface-based factorization procedure described in chapter 9 to recover the 3D shape and 3D motion parameters of the pedestal from the 2D motion parameters estimates.

We derived from the estimated 3D shape parameters the relative depth of the pedestal shown on the left image of Figure 11.7. In this image the brightness level of a pixel codes the relative depth of that pixel, the brighter the pixel, the closer it is to the camera in reference frame 1.

We extracted the texture of the pedestal from the video sequence by proceeding as detailed in the previous section. Then, we reconstructed the 3D shape and superimposed the texture. Two perspective views of the reconstructed 3D shape are shown on the center

and rightmost images of Figure 11.7. The nine planar patches of the pedestal are clearly seen as well as the angles between them. These two images represent two different views and are obtained by rotating the 3D model. Other views are generated in a similar way.



Figure 11.7: Relative depth and reconstructed 3D shape and texture for the pedestal video sequence of Figure 11.6.

11.3 Clown

This experiment illustrates the performance of the *multiresolution continuation-type method* developed in chapter 10. We used a sequence of 10 frames from a real life video sequence showing a toy clown. Figure 11.8 shows frames 1 and 5 of the clown video sequence. Each frame has 384×288 pixels.

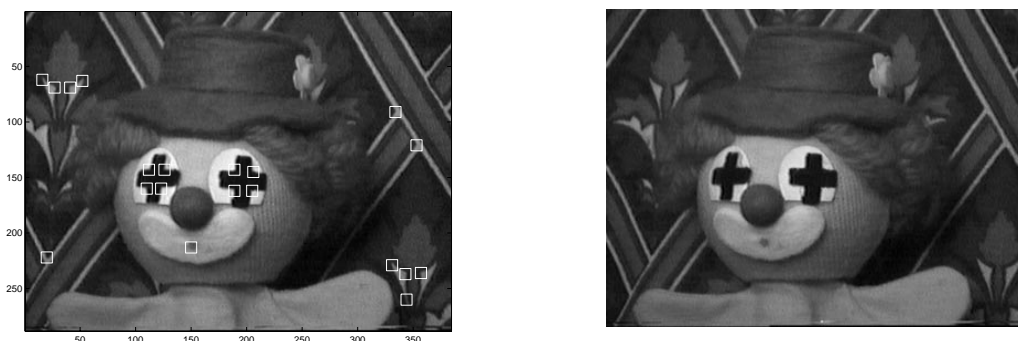


Figure 11.8: Clown video sequence. Frames 1 and 5.

Superimposed on frame 1, we marked with white squares 20 feature points used in

the *rank 1 factorization method*. The feature points were selected by using the intensity gradient criteria developed in chapter 3.

We tracked the feature points by matching the intensity pattern of each feature along the sequence, as described in chapter 3. Using the rank 1 factorization method described in chapter 8, we recovered the 3D motion from the feature trajectories.

The 3D shape estimate provided by the feature-based factorization method, i.e., the 3D shape described by the estimated 3D positions of the feature points is not a good approximation of the true 3D shape because the set of features is very sparse.

To recover a dense representation of the 3D shape, we estimated the relative depth of the 3D object by using the *multiresolution continuation-type method* described in chapter 10.

The evolution of the estimate of the relative depth is illustrated by Figure 11.9. The grey level images in this figure code the relative depth estimates. The brighter a pixel is, the closer to the camera it is in the first frame. The size of the estimation region \mathcal{R} was 30×30 pixels when processing the first 3 frames, 20×20 pixels when processing frames 4 to 6, and 10×10 pixels when processing frames 7 to 10. The left image was obtained with the first three frames and shows a very coarse estimate of the shape. The right image was obtained after all 10 frames of the image sequence have been processed.

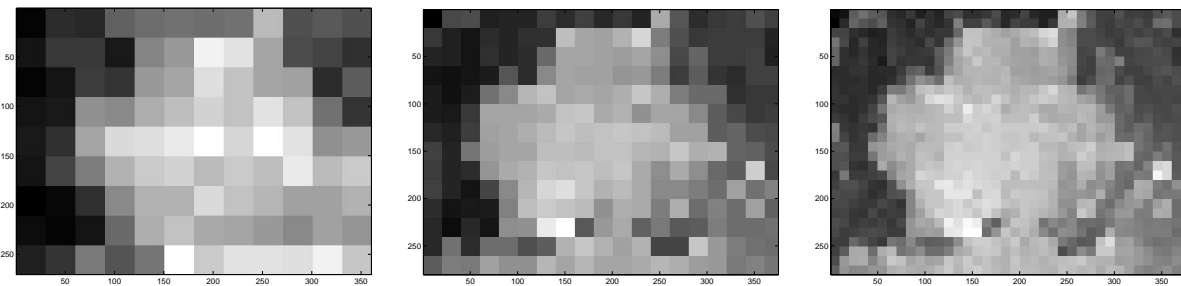


Figure 11.9: Relative depth recovered from the clown video sequence of Figure 11.8 after processing 3, 6, and 10 frames.

Chapter 12

Conclusion

This chapter concludes the thesis. We summarize the content of the thesis, emphasize the major original contributions, and point out future research directions.

12.1 Thesis Summary

The thesis addresses two problems within motion analysis: the segmentation of a two-dimensional (2D) rigid moving object, Part I, chapters 2 through 6; and the inference of three-dimensional (3D) rigid structure from a monocular video sequence, Part II, chapters 7 through 11. The following paragraphs summarize the contents of each chapter of the thesis.

Part I

In chapter 2 we stated the 2D motion segmentation problem. The 2D shape of the moving object was described by a discretized binary template. Each pixel of each frame in the video sequence was modeled as a noisy version either of the moving object intensity level (object texture), if that pixel belongs to the template of the moving object, or of the background intensity level (background texture), if otherwise. This model accounts explicitly for the occlusion of the background by the moving object over a set of frames. By modeling occlusion, we are able to obtain an accurate estimate of the moving object template, even when processing low texture / low contrast video sequences. The seg-

mentation problem was formulated as the *Maximum Likelihood* (ML) estimation of all the unknowns from the video sequence: the motions of the object, the motions of the camera, the binary template of the object, the texture of the object, and the texture of the background.

We minimized the ML cost function by first decoupling the estimation of the motions from the estimation of the remaining parameters. The motions were estimated on a frame by frame basis. In chapter 3 we dealt with the estimation of the motion of the brightness pattern in the image plane. We determined the estimation error variance of the motion parameters in terms of the spatial variability of the brightness pattern and the size of the region of analysis. In Part II, we used these expressions to weight the contribution of each image region to the estimation of 3D rigid structure from 2D motion.

In chapter 4 we detailed our two-step iterative algorithm that segments the 2D rigid moving object. We first estimated the texture of the moving object. Then, we inserted this texture estimate into the ML cost function, being left with a function of the texture of the background and of the template of the moving object. We described an algorithm to minimize this cost function iteratively in two steps: (i) estimate the background texture with known template; and (ii) estimate the object template with known background. The background estimate in step (i) was obtained in closed-form. The template estimate in step (ii) lead to a simple binary test evaluated at each pixel. The template estimation test integrates over time the (possibly small) intensity differences between the background and the moving object. We illustrated the good quality of the template estimates by segmenting a difficult sequence when a complex shaped object moves against a low contrast background.

In chapter 5 we analyzed statistically the binary test involved in the two-step iterative algorithm. We derived upper bounds for the probability of misclassifying the template pixels, and showed that these upper bounds decrease with the number of frames processed. We illustrated the convergent behavior of the test by segmenting a 1D object whose template is the union of two disjoint segments.

Chapter 6 tests our segmentation algorithm with two real video sequences. The first records a live traffic scene. The background moves, due to camera panning motion, and a number of moving cars enter and leave the scene. The cars have regions with very low texture, and the contrast of the cars with respect to the background is low in several regions. Our algorithm segments accurately the moving cars, and reconstructs successfully a complete view of the background. The second sequence is a robot soccer video clip. It shows a square shaped robot following a ball. The robot and the ball move against a static background, the field of the game, which has stripes that have intensity level perceptually similar to the robot's intensity level (zero contrast). This experiment illustrates the time evolution of the template estimate when the contrast between the moving object and the background is very low. Again, the algorithm successfully segments both the ball and the robot.

Part II

In chapter 7 we stated the problem of inferring 3D structure from an image sequence. We modeled the video frames as the orthogonal projection of the texture of the object on the image plane plus noise. Inferring 3D structure was formulated as the joint ML estimation of the 3D shape of the object, of its 3D motion, and of its texture. We reduce the dimensionality of the problem by minimizing first the ML cost function with respect to the object texture. This leads to an ML cost function that depends only on the 3D shape and 3D motion through the 2D motion induced in the image plane. This shows that the usual *structure from motion* (SFM) approach is an approximation to the ML solution.

Chapters 8 and 9 dealt with recovering structure from motion. In chapter 8 we developed a *rank 1 factorization* algorithm. A set of features is tracked across a set of frames. The 3D shape is represented by the set of 3D positions of the feature points. The shape of the trajectory of the projection of each feature point depends both on the depth of the point and the 3D motion of the object. The challenge in recovering 3D structure from 2D motion is to analyze this coupled dependence. Our rank 1 factorization method de-

couples the influence of the 3D shape from the influence of the 3D motion on the feature trajectories. In fact, by using an appropriate linear subspace projection, we obtained a new set of trajectories whose shape depends only on the 3D motion (the depth of each feature point determines the magnitude of the corresponding trajectory). The subspace projection enabled us to recover the 3D shape and the 3D motion from the set of feature trajectories by a simple factorization of a matrix that is rank 1 in a noiseless situation. This matrix was factored by computing the singular vector corresponding to the largest singular value, avoiding the use of the *Singular Value Decomposition* (SVD) as suggested by Tomasi and Kanade for their original *factorization method* [59, 61]. This led to a very fast algorithm to recover 3D structure from 2D motion, even when using a large number of features and large number of frames. We showed that the computational cost of our rank 1 factorization is significantly lower than the original factorization algorithm in [59, 61]. For the example tested, the computational cost is reduced by a factor of approximately 20.

We noted that when the goal is the recovery of a dense representation of the 3D shape, the feature-based approach may not solve the problem satisfactorily because it may require tracking an excessive number of features to obtain a dense description of the 3D shape. This leads to a difficult correspondence problem since only distinguished points, as brightness corners, can be accurately tracked. To overcome this limitation, we introduced in chapter 9 a new methodology: the *surface-based factorization*. First, we represented the 3D shape by a parametric description of the object surface. Then, we showed how to recover the parameters that describe the 3D shape and the 3D motion from the parameters describing the 2D motion in the image plane. We accomplished this by factorizing a *surface-based measurement matrix* that collects the set of 2D motion parameters. To factorize this matrix in an efficient way, we used the methodology of the rank 1 factorization developed in chapter 8. In chapter 9 we also showed how to make the factorization methods more robust by weighting differently the 2D motion parameters according to the accuracy of their estimates. The weights are computed from the variances of the estimates

of the 2D motion parameters. These variances were evaluated as detailed in chapter 3. We extend the rank 1 factorization method to the *weighted factorization method*. The weighted factorization recovers the 3D structure using the confidence weights without paying an additional computational cost. This was achieved by rewriting the problem with weights as the factorization of a modified measurement matrix and factorizing this matrix with the rank 1 factorization method. Chapter 9 includes one experiment that demonstrated the good performance of the surface-based factorization method when recovering rigid structure from an image sequence with a challenging smooth texture for which it is not possible to track pointwise features. A second experiment illustrated the improvement the quality of the reconstruction of the 3D structure, when the estimation weights are taken into account. The weighted estimates of the 3D structure are more accurate than the non-weighted ones because the 2D motion estimates corresponding to regions with higher spatial variability are given higher credit than the ones corresponding to regions with low texture.

In chapters 8 and 9 we approximated the ML estimation of the 3D structure by recovered the 3D shape and the 3D motion from the 2D motion of the brightness pattern in the image plane. The quality of the estimate of the 3D shape obtained this way is limited by the possibility of estimating the 2D motion over the entire image. For example, with image sequences with textures that exhibit a predominant spatial orientation, it is very difficult to compute the 2D motion of the brightness pattern (this problem was studied in detail in chapter 3). For these sequences, the 3D shape reconstruction obtained through the structure from motion approach may be much rougher than the corresponding ML estimate. In chapter 10 we proposed a method to estimate the 3D shape directly from the image intensity values by minimizing the ML cost function, after introducing the estimates of the 3D motions. To minimize the ML cost function, we developed a *multiresolution* scheme – a *continuation-type* algorithm – that works by estimating coarse approximations to the 3D shape at the beginning, and refining the estimate as more images are being taken into account. Each stage of the continuation algorithm uses a

very simple *Gauss-Newton method* to update the estimate of the 3D shape. We presented one experiment that illustrated the good convergence of our method when recovering a dense estimate of the 3D shape.

Chapter 11 describes three experiments with real video sequences. The first two experiments use the *surface-based rank 1 factorization method* developed in chapters 8 and 9 to recover 3D structure from video sequences. The box experiment uses a video sequence obtained by rotating a camera around a box with rectangular faces that stands over a carpet. Our algorithm recovered the 3D shape of the scene as a piecewise planar surface with patches corresponding to the faces of the box and the floor. Once the 3D structure is available, we can recreate new views of the scene, possibly not present in the original sequence. The chapter includes two such views synthesized by specifying different viewing positions for the camera. The correctness of the angles between the planar patches in the recovered surface demonstrate the good performance of our method to recover 3D rigid structure from motion. We used the box video sequence to illustrate how the 3D model-based video representation achieves very high compression rates. We presented reasonably good quality frames obtained from the box sequence compressed at ratios 317:1 and 575:1. Chapter 11 includes a second experiment that also demonstrates the performance of the surface-based factorization method in recovering piecewise planar shapes. The last experiment uses a video clip that shows a clown toy as an example of a scene whose 3D shape is not well described by a sparse set of features. We first recovered the 3D motion by using the feature-based *rank 1 factorization method*. Then, we used the *multiresolution continuation-type* algorithm developed in chapter 10 to recover the 3D shape directly from the image intensity values. The depth map that is recovered shows this algorithm is able to provide a dense representation of general 3D rigid shapes.

12.2 Major Contributions

In this section we summarize our major contributions. The thesis presented an original *Maximum Likelihood* (ML) formulation to the problem of inferring rigid structure from a set of images. In the context of this formulation, we developed efficient algorithms to segment a two-dimensional (2D) rigid moving object, and to recover three-dimensional (3D) rigid structure from video. We detail these accomplishments next.

Inference of 3D rigid structure

Our major contributions to the recovering 3D rigid structure from 2D videos are the *surface-based factorization* framework with its associated *rank 1 factorization* and *weighted factorization* algorithms, and the *multiresolution continuation-type method* that refines the 3D shape estimate directly from the image intensity values. They are summarized as follows:

- **Surface-based factorization.** We introduce the *surface-based factorization method* that represents the surfaces to be reconstructed by parametric models, and recovers, by a suitable factorization procedure, the 3D structure by an inversion from a parsimonious set of 2D motion parameters. These describe the motions of the brightness pattern in the image plane induced by the three dimensional motions of the rigid object. Our approach overcomes the known difficulty of tracking pointwise features as required in the original factorization method of Tomasi and Kanade [59, 61].
- **Rank 1 factorization.** We exploit the constraints of the 3D structure recovery from 2D motions to derive a fast *rank 1 factorization* algorithm. This algorithm avoids the factorization of a rank 3 matrix as in [59, 61].
- **Weighted factorization.** We developed the *weighted factorization method* that weights differently motion estimates of different error quality. The weighted factorization has the same computational cost of the non-weighted rank 1 factorization.

- **Direct inference of 3D rigid shape.** We develop a new algorithm that recovers 3D rigid shape directly from the image intensity values through *Maximum Likelihood* estimation. The direct inference method progressively refines the estimates of the 3D shape in a coarse-to-fine *multiresolution continuation-type* manner. Each step of this continuation procedure is solved by a *Gauss-Newton method* that, in practice, requires no more than two or three iterations.

Segmentation of 2D moving object

We model explicitly the occlusion of the background by the moving object. By considering the occlusion, we can segment accurately even low texture / low contrast video sequences. We present a computationally simple algorithm that computes the ML estimate of the template of the moving object. We summarize this accomplishment briefly.

- **Explicit modeling of occlusion.** The model considers explicitly the occlusion of the background by the moving object.
- **Motion segmentation in low texture / low contrast.** Our new algorithm resolves the difficulties that arise when processing low texture and/or low contrast videos. The robustness of the algorithm is due to two important characteristics: it uses a set of frames, rather than a single pair; and it processes directly the image intensity values, through ML estimation, rather than relying on the computation of the image motion field as an intermediate step. The algorithm is iterative alternating between estimating the background intensity levels and the moving object template. Both steps have closed-form solution and are computationally very simple.

12.3 Future Directions

We highlight a few alternative directions to explore our solution to the problem of rigid structure recovery from monocular videos. We group them into three categories: direct

extensions to our work; new directions that lead to new research problems; and applications.

Direct extensions of our work

Our formulation of the *Maximum Likelihood* (ML) may be explored in a number of ways to recover rigid structure from video, including the following ones.

Segmentation of 2D moving object – parameterized contour model. We described by a discretized binary template the two-dimensional (2D) shape of the moving object. This description is highly flexible and enables the accurate detection of non-smooth and complex shapes. Its drawback is that to estimate the binary template requires that the background (including occluded regions in some frames) be completely seen in the video sequence. If the sequence is not long enough or the motion of the object is such that some portion of the background remains occluded, the algorithm does not recover the object template. In this case, it is computationally very expensive to minimize the ML cost function as given by expressions (4.8-4.11) in chapter 4, section 4.3, by using generic optimization methods. The number of unknowns that describe the template is prohibitively high. One way to overcome this problem is to parameterize the 2D shape of the object by a small number of parameters. A contour-based description of the object boundary by Fourier Descriptors [33] or Splines [24] for example fits this need. A possible strategy is to replace the background estimate given by expression (4.7) in the ML cost function, equations (4.8-4.11), and to minimize the resulting functional with respect to the parameters that describe the contour of the moving object.

Direct inference of 3D rigid structure – parameterized surface model. We recover the three-dimensional (3D) shape directly from the image intensity values by estimating a dense depth map. Possible extensions include the investigation of different 3D shape models. Parametric models enable more compact and robust shape representation. The ML estimate can be computed by using the same strategy we used – introduce the estimate of the 3D motion into the ML cost function of expression (7.10), then min-

imize it with respect to the 3D shape. Estimation of the 3D structure directly from the image intensity values, through ML estimation, provides a robust way of dealing with such issues as segmentation, for piecewise shape models, and model complexity, by using classic tools such as Bayesian inference or information-theoretic criteria [12].

Surface-based rank 1 factorization extensions. In the thesis, we model the video frames as orthogonal projections of the object texture. The *surface-based rank 1 factorization method* can be extended to more general models like the scaled-orthography and the para-perspective models that approximate better the perspective projection when the orthographic model is not valid, see references [47, 48, 49] for the extension of the original factorization method of Tomasi and Kanade [59, 60, 61]. Another extension of our methodology would be dealing with multiple moving objects, see references [19, 20, 21] for a feature-based factorization method for the multibody scenario.

New research directions

Inferring automatically general 3D structure from 2D video is an active area of research. The work described in the thesis inspires two new directions that may be explored in the future.

Subspace constraints for image motion estimation. In the thesis we use linear subspace constraints for efficient motion analysis, i.e., to recover the 3D structure from the 2D motion in the image plane in an expedite way. The same type of constraints can be used to improve the accuracy of the 2D motion estimates. The estimation of the 2D motion of the brightness pattern in the image plane is a key step in recovering the 3D structure. The texture of the brightness pattern, the size of the 2D displacements, and noise, may conspire to make the task of tracking features or regions very difficult. One reason for this difficulty is that the 2D motion for each region is estimated in a local way, i.e., independently of the other regions. But the parameters describing the 2D motion of the different regions are not unrelated, due to the constraints imposed by the rigidity of the scene. To improve the accuracy of the estimation of the 2D motion,

recent research work makes use of linear subspace methods. References [68, 69] use linear subspace constraints to estimate the 2D motion parameters that best align a set of planes. We show in the thesis that the space of the matrices $\tilde{\mathbf{R}}$ that collect 2D motion parameters is highly restricted – matrix $\tilde{\mathbf{R}}$ is rank 1 in a noiseless situation (expression (8.21)). The use of this severe constraint in estimating the 2D motion of the brightness pattern in a global way should be investigated.

Subspace constraints for motion analysis. Recently, subspace constraints have been used in several tasks within motion analysis, including the handling of scenes with self-occlusion, see reference [43], and the recovery of 3D deformable shape from a set of cameras, see reference [56]. The impact of the rank 1 constraint on these tasks should be investigated.

Recovery of complete 3D models. The majority of current 3D modeling methods work in a kind of open loop fashion – the problem has been stated as “given a set of images that show roughly the same portion of the scene, estimate the underlying 3D structure”. In general, due to the occlusion and the limited field of view, the entire 3D scene we want to recover is not completely seen in that set of images. The 3D models obtained are thus very incomplete. An important research direction is to seek expedite ways to close the loop, i.e., to enable the assimilation of larger sets of images. This framework would enable the automatic recovery of complete 3D scenes that are only partially observed in each image of the video sequence. Obviously, when the scene contains strong perspective distortions, the perspective projection must be taken into account. The expedite recovery of 3D structure for this kind of scenes, with the associated problems, like camera self-calibration and model selection, must be further investigated.

Applications

We highlight below some potential applications of the algorithms developed in the thesis.

Virtualized reality Current methods to generate virtual scenarios are expensive. Either 3D models are generated in a manual way which requires a human to specify the details

of the 3D shape and texture, or auxiliary equipment like a laser range finder need to be used to capture the 3D reality. Our work can be used to generate automatically virtual reality scenes. The 3D models obtained from the real life video data can be used to build synthetic image sequences. The synthesis is achieved by specifying the sequence of viewing positions along time. The viewing positions are arbitrary – they are specified by the user, either in an interactive way, or by an automatic procedure. More complex scenes can be obtained by merging real objects with virtual entities.

Efficient representations for video. Multiple global motion models and long-term prediction proved to reduce video compression ratios, see references [53, 67]. The use of 3D models further reduces the amount of data needed to code a video sequence, see for example references [26, 42]. When there is no prior knowledge about the scene, a video representation scheme based in the 3D models recovered through *surface-based factorization* can be used. Instead of representing a video sequence in terms of frames and pixels, we can use the recovered 3D structure. A video sequence is then represented by the 3D shape and texture of the object, and its 3D motion. Since the 3D motion and 3D shape are coded with a few parameters, the number of bytes necessary to code the entire sequence is governed by the size of the object texture representation. The texture is coded as a set of ordinary images, one for each planar patch. By using this model-based representation, we reduce dramatically the storage space because we code only once the brightness values, as opposed to the redundancy of coding the brightness values at each of the frames of the original sequence.

Other applications of 3D model-based video representations include content-based addressing. Current systems that provide content-based access work by first segmenting the video in a sequence of shots and then labeling each shot with a distinctive indexing feature. The most common features used are image-based features, such as color histograms or image moments. By using 3D models we improve both the temporal segmentation and the indexing. The temporal segmentation can account for the 3D content of the scene. Indexing by 3D features, directly related to the 3D shape, enable queries by object

similarity. See reference [42] for illustrative examples of the use of 3D models in video processing.

Appendix A

Moments of Functions of Random Variables

Consider a random vector \mathbf{v} and a vector function $\mathbf{f}(\mathbf{v})$. In this appendix we present expressions for the first-order approximation of the mean and covariance of the random vector $\mathbf{f}(\mathbf{v})$ in terms of the mean and covariance of \mathbf{v} and the partial derivatives of $\mathbf{f}(\cdot)$. These results are known from estimation theory, see for example reference [46].

A.1 First-order approximation of the mean

We denote by $\bar{\mathbf{v}}$ the mean value of the random vector \mathbf{v} ,

$$\bar{\mathbf{v}} = \text{E} \{ \mathbf{v} \} . \quad (\text{A.1})$$

The first-order approximation of the moments of the random vector $\mathbf{f}(\mathbf{v})$ is obtained by neglecting the second and higher order terms of the Taylor series expansion of $\mathbf{f}(\cdot)$ expanded around the mean value $\bar{\mathbf{v}}$ of \mathbf{v} , i.e.,

$$\mathbf{f}(\mathbf{v}) \simeq \mathbf{f}(\bar{\mathbf{v}}) + \nabla_{\mathbf{v}} \mathbf{f}(\bar{\mathbf{v}}) (\mathbf{v} - \bar{\mathbf{v}}), \quad (\text{A.2})$$

where $\nabla_{\mathbf{v}} \mathbf{f}(\bar{\mathbf{v}})$ denotes the gradient of $\mathbf{f}(\cdot)$ evaluated at $\mathbf{v} = \bar{\mathbf{v}}$.

The mean value of $\mathbf{f}(\mathbf{v})$ is then approximated by

$$\text{E} \{ \mathbf{f}(\mathbf{v}) \} = \text{E} \{ \mathbf{f}(\bar{\mathbf{v}}) \} + \text{E} \{ \nabla_{\mathbf{v}} \mathbf{f}(\bar{\mathbf{v}}) (\mathbf{v} - \bar{\mathbf{v}}) \} . \quad (\text{A.3})$$

The quantities $\mathbf{f}(\bar{\mathbf{v}})$ and $\nabla_{\mathbf{v}}\mathbf{f}(\bar{\mathbf{v}})$ in expression (A.3) are deterministic. By using expression (A.1), we get $E\{\mathbf{v} - \bar{\mathbf{v}}\} = \mathbf{0}$, thus, expression (A.3) leads to

$$E\{\mathbf{f}(\mathbf{v})\} = \mathbf{f}(\bar{\mathbf{v}}). \quad (\text{A.4})$$

The first-order approximation of the mean value of the random vector $\mathbf{f}(\mathbf{v})$ is then the value of the vector function $\mathbf{f}(\cdot)$ evaluated at the mean value $\bar{\mathbf{v}}$ of \mathbf{v} .

A.2 First-order approximation of the covariance

The covariance matrix Σ_v of the random vector \mathbf{v} is defined as

$$\Sigma_v = E\{(\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^T\}. \quad (\text{A.5})$$

Analogously, the covariance matrix Σ_f of $\mathbf{f}(\mathbf{v})$ is given by

$$\Sigma_f = E\left\{\left(\mathbf{f}(\mathbf{v}) - E\{\mathbf{f}(\mathbf{v})\}\right)\left(\mathbf{f}(\mathbf{v}) - E\{\mathbf{f}(\mathbf{v})\}\right)^T\right\}. \quad (\text{A.6})$$

The first-order approximation of the covariance matrix Σ_f is obtained by using the truncated Taylor series approximation for $\mathbf{f}(\cdot)$ as given by expression (A.2). By inserting expression (A.2) into expression (A.6) and using expression (A.4) for $E\{\mathbf{f}(\mathbf{v})\}$, we obtain

$$\Sigma_f = E\left\{\nabla_{\mathbf{v}}\mathbf{f}(\bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^T\nabla_{\mathbf{v}}\mathbf{f}(\bar{\mathbf{v}})^T\right\}. \quad (\text{A.7})$$

Since $\nabla_{\mathbf{v}}\mathbf{f}(\bar{\mathbf{v}})$ is deterministic, and using the definition (A.5), we obtain the following expression that relates the covariance matrix of $\mathbf{f}(\mathbf{v})$ with the covariance matrix of \mathbf{v} and the gradient of the vector function $\mathbf{f}(\cdot)$,

$$\Sigma_f = \nabla_{\mathbf{v}}\mathbf{f}(\bar{\mathbf{v}})\Sigma_v\nabla_{\mathbf{v}}\mathbf{f}(\bar{\mathbf{v}})^T. \quad (\text{A.8})$$

To make explicit the contribution of each of the components of the random vector \mathbf{v} to the covariance matrix of $\mathbf{f}(\mathbf{v})$, we rewrite expression (A.8) in terms of a sum of cross terms. The first-order approximation of the covariance matrix of $\mathbf{f}(\mathbf{v})$ is then given by

$$\Sigma_f = \sum_{k,l \in V} E\left\{(v_k - \bar{v}_k)(v_l - \bar{v}_l)\right\} \frac{\partial \mathbf{f}}{\partial v_k}(\bar{\mathbf{v}}) \frac{\partial \mathbf{f}}{\partial v_l}(\bar{\mathbf{v}})^T, \quad (\text{A.9})$$

where $\{v_i, i \in V\}$ are the components of \mathbf{v} , $\overline{v_i}$ denotes the mean value of v_i , and $\frac{\partial \mathbf{f}}{\partial v_i}(\overline{\mathbf{v}})$ denotes the partial derivative of $\mathbf{f}(\cdot)$ with respect to v_i , evaluated at $\mathbf{v} = \overline{\mathbf{v}}$.

Bibliography

- [1] Pedro M. Q. Aguiar and José M. F. Moura. Incremental motion segmentation in low texture. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'96)*, Lausanne, Switzerland, September 1996.
- [2] Pedro M. Q. Aguiar and José M. F. Moura. Detecting and solving template ambiguities in motion segmentation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'97)*, Santa Barbara, CA, USA, October 1997.
- [3] Pedro M. Q. Aguiar and José M. F. Moura. Robust structure from motion under orthography. In H. Niemann, H.-P. Seidel, and B. Girod, editors, *Proceedings of the IEEE International Workshop on Image and Multidimensional Digital Signal Processing*, Alpbach, Austria, July 1998.
- [4] Pedro M. Q. Aguiar and José M. F. Moura. Video representation via 3D shaped mosaics. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*, Chicago, IL, USA, October 1998.
- [5] Pedro M. Q. Aguiar and José M. F. Moura. Factorization as a rank 1 problem. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 178–184, Fort Collins, CO, USA, June 1999.
- [6] Pedro M. Q. Aguiar and José M. F. Moura. Fast 3D modelling from video. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, Copenhagen, Denmark, September 1999.

- [7] Pedro M. Q. Aguiar and José M. F. Moura. A fast algorithm for rigid structure from image sequences. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, volume 3, pages 125–129, Kobe, Japan, October 1999.
- [8] Pedro M. Q. Aguiar and José M. F. Moura. Maximum likelihood inference of 3D structure from image sequences. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, number 1654 in Lectures Notes in Computer Science. Springer-Verlag, July 1999.
- [9] Pedro M. Q. Aguiar and José M. F. Moura. Weighted factorization. Submitted for publication, January 2000.
- [10] Nicholas Ayache. *Artificial Vision for Mobile Robots*. The MIT Press, Cambridge, MA, USA, 1991.
- [11] Serge Ayer. *Sequential and Competitive Methods for Estimation of Multiple Motions*. PhD thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 1995.
- [12] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, October 1998.
- [13] James R. Bergen, Patrick Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, number 588 in Lectures Notes in Computer Science, pages 237–252, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.
- [14] Åke Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.
- [15] Ephraim J. Borowski and J. M. Borwein. *The Harper Collins Dictionary of Mathematics*. HarperCollins Publishers, third edition, 1991.

- [16] Patrick Bouthemy and Edouard François. Motion segmentation and qualitative dynamic scene analysis from an image sequence. *Kluwer International Journal of Computer Vision (IJCV)*, 10(2):157–182, April 1993.
- [17] Ted J. Broida and Rama Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(6):497–513, June 1991.
- [18] Michael M. Chang, A. Murat Tekalp, and M. Ibrahim Sezan. Simultaneous motion estimation and segmentation. *IEEE Transactions on Image Processing*, 6(9):1326–1333, September 1997.
- [19] João P. Costeira. *A Factorization Method for Independently Moving Objects*. PhD thesis, Instituto Superior Técnico, Lisboa, Portugal, September 1996.
- [20] João P. Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'95)*, Cambridge, MA, USA, June 1995.
- [21] João P. Costeira and Takeo Kanade. A factorization method for independently moving objects. *Kluwer International Journal of Computer Vision (IJCV)*, 29(3):159–179, September 1998.
- [22] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the SIGGRAPH International Conference on Computer Graphics and Interactive Techniques*, August 1996.
- [23] Norbert Diehl. Object-oriented motion estimation and segmentation in image sequences. *Signal Processing: Image Communication*, 3(1):23–56, February 1991.
- [24] Paul Dierckx. *Curve and Surface Fitting with Splines*. Monographs in Numerical Analysis. Oxford Science Publications, 1995.

- [25] Marie-Pierre Dubuisson and Anil K. Jain. Contour extraction of moving objects in complex outdoor scenes. *Kluwer International Journal of Computer Vision (IJCV)*, 14(1):83–105, January 1995.
- [26] Bernd Girod, Peter Eisert, Marcus Magnor, Eckehard Steinbach, and Thomas Wiegand. 3D image models and compression: Synthetic hibrid or natural fit ? In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, volume 3, Kobe, Japan, October 1999.
- [27] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins Series in Mathematical Sciences. The Johns Hopkins University Press, second edition, 1989.
- [28] Joachim Heel. Direct estimation of structure and motion from multiple frames. MIT AI Lab. Memo 1190, Massachussets Institute of Technology, MA, USA, March 1990.
- [29] Berthold K. P. Horn and E. J. Weldon Jr. Direct methods for recovering motion. *Kluwer International Journal of Computer Vision (IJCV)*, 2(1):51–76, June 1988.
- [30] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
- [31] Michal Irani and Shmuel Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communications and Image Representation*, 4(4):324–335, December 1993.
- [32] Michal Irani, Benny Rousso, and Shmuel Peleg. Computing occluding and transparent motions. *Kluwer International Journal of Computer Vision (IJCV)*, 12(1):5–16, February 1994.
- [33] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall Information and Sciences Series. Prentice-Hall International Inc., 1989.
- [34] Radu S. Jasinschi. *Generative Video: A Meta Video Representation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, September 1995.

- [35] Radu S. Jasinschi and José M. F. Moura. Content-based video sequence representation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'95)*, volume 2, Washington D.C., USA, September 1995.
- [36] Radu S. Jasinschi and José M. F. Moura. *Generative Video: Very Low Bit Rate Video Compression*. U.S. Patent and Trademark Office, S.N. 5,854,856, issued December 29, 1998.
- [37] Radu S. Jasinschi, José M. F. Moura, Jia-Ching Cheng, and Amir Asif. Video compression via constructs. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'95)*, volume 4, Detroit, MI, USA, May 1995.
- [38] David C. Knill, Daniel Kersten, and Alan Yuille. A Bayesian formulation of visual perception. In *Perception as Bayesian Inference*, pages 1–21. Cambridge University Press, 1996.
- [39] Haibo Li, Astrid Lundmark, and Robert Forchheimer. Image sequence coding at very low bitrates: A review. *IEEE Transactions on Image Processing*, 3(5):589–608, September 1994.
- [40] I-Jong Lin and S. Y. Kung. Automatic video object segmentation via Voronoi ordering and surface optimization. In *Proceedings of the IEEE International Workshop on Multimedia SignalProcessing*, pages 265–270, Copenhagen, Denmark, September 1999.
- [41] Wolfgang Luth. Segmentation of image sequences. *Theoretical Foundations of Computer Vision, Mathematical Research – Akademie Verlag*, pages 215–226, 1992.
- [42] Fernando C. M. Martins and José M. F. Moura. Video representation with three-dimensional entities. *IEEE Journal on Selected Areas in Communications*, 16(1):71–85, January 1998.

- [43] Minoru Maruyama and Satoshi Kurumi. Bidirectional optimization for reconstructing 3D shape from an image sequence with missing data. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, volume 3, Kobe, Japan, October 1999.
- [44] Toshihiko Morita and Takeo Kanade. A sequential factorization method for recovering shape and motion from image streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(8):858–867, August 1997.
- [45] Joseph A. O'Sullivan, Richard E. Blahut, and Donald L. Snyder. Information-theoretic image formation. *IEEE Transactions on Information Theory*, 44(6):2094–2123, October 1998.
- [46] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. Electrical & Electronic Engineering Series. McGraw Hill, third edition, 1991.
- [47] Conrad J. Poelman. *A Paraperspective Factorization Method For Shape and Motion Recovery*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.
- [48] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. In *Proceedings of the European Conference on Computer Vision*, number 801 in Lectures Notes in Computer Science, pages 97–108, Stockholm, Sweden, May 1994. Springer-Verlag.
- [49] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(3):206–218, March 1997.
- [50] H. Shum, K. Ikeuchi, and R. Reddy. Virtual reality modeling from a sequence of range images. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Germany, September 1994.

- [51] S. Soatto, P. Perona, R. Frezza, and G. Picci. Recursive motion and structure estimations with complete error characterization. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 428–433, New York City, NY, USA, June 1993.
- [52] Marc Soucy and Denis Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(4):344–358, April 1995.
- [53] Eckehard Steinbach, Thomas Wiegand, and Bernd Girod. Using multiple global motion models for improved block-based video coding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, volume 2, Kobe, Japan, October 1999.
- [54] Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 261–272. Springer Verlag, Berlin, 1999.
- [55] R. Szeliski and S. Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1), 1994.
- [56] Joo Kooi Tan and Seiji Ishikawa. Recovering entire shape of a deformable object employing a single measurement matrix. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, volume 3, Kobe, Japan, October 1999.
- [57] A. Murat Tekalp. *Digital Video Processing*. Prentice Hall Signal Processing Series. Prentice Hall, 1995.
- [58] J. Inigo Thomas, Allen Hansen, and John Oliensis. Understanding noise: The critical role of motion error in scene reconstruction. In *Proceedings of the IEEE International*

- Conference on Computer Vision (ICCV'93)*, pages 325–329, Berlin, Germany, May 1993.
- [59] Carlo Tomasi. *Shape and Motion from Image Streams: a Factorization Method*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1991.
- [60] Carlo Tomasi and Takeo Kanade. Shape and motion without depth. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'90)*, Osaka, Japan, June 1990.
- [61] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *Kluwer International Journal of Computer Vision (IJCV)*, 9(2):137–154, November 1992.
- [62] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory, Part I: Detection, Estimation, and Linear Modulation Theory*. John Wiley & Sons, 1968.
- [63] Roger Y. Tsai and Thomas S. Huang. Estimating three-dimensional motion parameters of a rigid planar patch. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 29(6):1147–1152, December 1981.
- [64] Roger Y. Tsai and Thomas S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6(1):13–27, January 1984.
- [65] Shimon Ullman. *The Interpretation of Visual Motion*. MIT Press, Cambridge, MA, USA, 1979.
- [66] Manuela Veloso, Peter Stone, Sorin Achim, and Mike Bowling. A layered approach for an autonomous robotic soccer system. In *Proceedings of the First International Conference on Autonomous Agents*, Marina del Rey, CA, USA, February 1997.

- [67] Thomas Wiegand, Eckehard Steinbach, and Bernd Girod. Long-term memory prediction using affine motion compensation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, volume 1, Kobe, Japan, October 1999.
- [68] Lihi Zelnik-Manor and Michal Irani. Multi-frame alignment of planes. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 151–156, Fort Collins, CO, USA, June 1999.
- [69] Lihi Zelnik-Manor and Michal Irani. Multi-view subspace constraints on homographies. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'99)*, volume 2, pages 710–715, Kerkyra, Greece, September 1999.
- [70] Heyun Zheng and Steven D. Blostein. Motion-based object segmentation and estimation using the MDL principle. *IEEE Transactions on Image Processing*, 4(9):1223–1235, September 1995.