OBJECT SEGMENTATION IN IMAGE SEQUENCES USING MOTION AND COLOR INFORMATION

by

Yi Li

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright by Yi Li, 1999
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-45991-8
OBJECT SEGMENTATION IN IMAGE SEQUENCES USING MOTION AND COLOR INFORMATION

Yi Li

Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
1999

Abstract

Accurate segmentation of moving objects in image sequences is of paramount importance for many object based multimedia applications. In this thesis, we present a novel automatic, multi-frame, region-feature based object segmentation technique. It combines the advantages of feature based methods and gradient based methods. Salient region features are extracted from the first two frames of an image sequence and are tracked over a number of frames. Trajectory clustering is then performed to group these features into putative objects, from which a set of motion models are estimated. Final segmentation result is obtained by region classification based on these motion models. The proposed technique uses both static and motion information to precisely localize object boundaries. It provides reliable and coherent interpretation of the scene over time by exploiting temporal information from several frames. Experimental results on a variety of image sequences clearly show its advantages over traditional techniques.
Acknowledgments

As always, there are many people to thank. Foremost among them are Professor Dimitrios Hatzinakos and Professor Anastasios N. Venetsanopoulos, whom I thank not only for their wise supervision, continuous encouragement and ready accessibility, but also for affording me the freedom to develop my own research project and pursue the directions that intrigued me most.

I would like to thank my committee members, Professor George V. Eleftheriades, Professor Jinwoo Choe and Professor Kostas N. Plataniotis for carefully reviewing my thesis and providing useful suggestions.

This work was supported in part by Communication Information Technology of Ontario (CITO).

My gratitude extends to my friends and colleagues at University of Toronto. Thanks for all of the good times and assistance, both technical and non-technical.

I would like to thank Dr. W.Y. Ma at HP Labs and Professor B.S. Manjunath in the Department of Electrical and Computer Engineering at University of California at Santa Barbara for providing the binary code of the Edge Flow algorithm.

Finally, none of this would have been possible without the unfailing love and continuous encouragement I received from my parents and sister throughout my life.
Contents

Abstract ........................................... ii

Acknowledgements ................................ iii

List of Tables ........................................ vii

List of Figures ....................................... viii

1 Motivation ........................................... 1
   1.1 Object based multimedia ......................... 1
       1.1.1 Traditional video coding .................... 1
       1.1.2 Object based multimedia .................. 2
   1.2 Object extraction and motion based segmentation .. 6
       1.2.1 What is an object? ........................ 7
       1.2.2 Problem definition and assumption ........ 8
       1.2.3 Applications ............................ 9
   1.3 Previous work .................................. 10
   1.4 Contributions of the thesis ...................... 13
   1.5 Thesis outline ................................ 15

2 Region feature extraction .......................... 16
   2.1 Feature based motion analysis .................. 16
       2.1.1 Introduction ............................ 16
       2.1.2 The case for region features ............ 17
## Contents

2.2 Static image segmentation ........................................ 18
2.3 Region grouping .................................................. 20
2.4 Region feature correspondence ................................. 21
   2.4.1 Introduction ................................................. 22
   2.4.2 Feature correspondence based on region size .............. 22
   2.4.3 Rejecting miscorrespondence ................................ 25

3 Motion estimation and feature tracking .......................... 26
   3.1 Optical flow computation ..................................... 26
      3.1.1 Constant brightness constraint equation ............... 27
      3.1.2 Coarse-to-fine strategy .................................. 30
      3.1.3 Limitations ................................................. 31
   3.2 Region based parametric motion estimation .................... 32
      3.2.1 Geometric transformation ................................. 32
      3.2.2 Affine motion model ...................................... 34
      3.2.3 Affine motion estimation .................................. 35
      3.2.4 Limitations ................................................. 37
   3.3 Motion estimation with initial guess .......................... 40
      3.3.1 Initial guess .............................................. 40
      3.3.2 Implementation of the "affine flow" technique .......... 41
      3.3.3 Robust motion estimation ................................. 44
   3.4 Feature tracking .............................................. 45

4 Feature clustering and region classification .................... 49
   4.1 Feature clustering ............................................. 49
      4.1.1 Clustering concepts ...................................... 50
      4.1.2 Trajectory clustering using singular value decomposition ... 51
      4.1.3 Motion model extraction .................................. 57
   4.2 Region classification .......................................... 58
      4.2.1 Motion model assignment .................................. 58
      4.2.2 Bidirectional region warping .............................. 59
Contents

5 Results and conclusions 62
  5.1 Simulation results .............................................. 62
    5.1.1 The synthetic Two-bars sequence ......................... 62
    5.1.2 The Table Tennis sequence ................................. 64
    5.1.3 The Calendar sequence ................................... 70
    5.1.4 The Flower Garden sequence ............................... 76
    5.1.5 The Foreman sequence ................................... 79
  5.2 Performance and complexity .................................. 83
    5.2.1 Performance ............................................... 83
    5.2.2 Complexity .............................................. 85
  5.3 Discussions and future directions ............................ 86
  5.4 Conclusions .................................................. 88

Bibliography 89
List of Tables

1.1 Three stages for the proposed object segmentation algorithm . . . . . 15
2.1 Three steps for feature extraction ........................................ 18
5.1 Parameter setting for the simulations ................................. 84
# List of Figures

1.1 An MPEG-4 scene .................................................. 4  
1.2 Sprite coding of MPEG-4 video sequence .......................... 5  

2.1 Edge Flow segmentation ........................................... 19  
2.2 Region feature extraction ........................................ 24  

3.1 Examples of geometric transformation ............................ 34  
3.2 The synthetic Two-bars sequence .................................. 39  
3.3 Feature trajectories of the synthetic Two-bars sequence .......... 47  

4.1 Regions along motion boundaries ................................... 60  
4.2 Region classification for the synthetic Two-bars sequence ... 60  

5.1 Simulation results for the synthetic Two-bars sequence ........... 63  
5.2 The MPEG Table Tennis sequence .................................... 65  
5.3 Static segmentation result for the Table Tennis sequence .......... 66  
5.4 Feature extraction and tracking results for the Table Tennis sequence 67  
5.5 Final segmentation result for the Table Tennis sequence .......... 68  
5.6 Object tracking result for the Table Tennis sequence .............. 69  
5.7 The MPEG Calendar sequence ....................................... 71  
5.8 Static segmentation result for the Calendar sequence .............. 72  
5.9 Feature extraction and tracking results for the Calendar sequence 73  
5.10 Final segmentation result for the Calendar sequence .............. 74  
5.11 Object tracking result for the Calendar sequence ................. 75  
5.12 The MPEG Flower Garden sequence ................................. 77
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.13</td>
<td>Static segmentation result for the Flower Garden sequence</td>
<td>77</td>
</tr>
<tr>
<td>5.14</td>
<td>Feature extraction and tracking results for the Flower Garden sequence</td>
<td>78</td>
</tr>
<tr>
<td>5.15</td>
<td>Final segmentation result for the Flower Garden sequence</td>
<td>78</td>
</tr>
<tr>
<td>5.16</td>
<td>Object tracking result for the Flower Garden sequence</td>
<td>79</td>
</tr>
<tr>
<td>5.17</td>
<td>The Foreman sequence</td>
<td>80</td>
</tr>
<tr>
<td>5.18</td>
<td>Static segmentation result for the Foreman sequence</td>
<td>80</td>
</tr>
<tr>
<td>5.19</td>
<td>Feature extraction and tracking results for the Foreman sequence</td>
<td>81</td>
</tr>
<tr>
<td>5.20</td>
<td>Final segmentation result for the Foreman sequence</td>
<td>81</td>
</tr>
<tr>
<td>5.21</td>
<td>Object tracking result for the Foreman sequence</td>
<td>82</td>
</tr>
</tbody>
</table>
Chapter 1

Motivation

1.1 Object based multimedia

1.1.1 Traditional video coding

In order to represent an image sequence with a minimal number of bits, it is necessary to perform efficient image coding. There are three interrelated parts in an image coding system: the encoding process, the representation, and the decoding process [1]. The choice of representation is the central determinant of the overall structure of the coding system. The encoding process converts the input image into the chosen representation, and the decoding process converts the representation into an array of pixels for the display.

Transform coding and subband coding are popular techniques that have been applied to the coding of still and moving images. In these techniques, an image is described as a sum of a set of basis functions, and this forms the implicit image representation. In the case of moving image sequences or video, these techniques are often combined with frame-differencing or motion compensation, to take advantage of the temporal structure of image sequences. The MPEG-2 video coding standard is such an example, in which case discrete cosine transform (DCT) is combined with block-matching to reduce the vast amount of data in image sequences.

The main advantage of transform based techniques (and subband based ones as
1.1 Object based multimedia

well) is that they involve a short and deterministic encoding and decoding delay. They are relatively low-risk methods, as they never fail totally given a reasonable bit rate. However, they also have two main limitations. First, in these techniques, compression inefficiency is traded for computation efficiency and regularity. Although they compress video data by taking out the enormous redundancy that occurs within and between frames and by taking advantage of the characteristics of human visual system, video as we know it still uses many more bits than necessary. Second, the bit stream generated by transform based techniques can not provide the flexibility and functionality that are required by many emerging multimedia applications.

The fundamental reason for these limitations is that the video representation used in the transform based techniques does not match very well the scene structures described by the video [2]. In DCT, for example, an image is represented as a sum of cosine basis functions which have nothing to do with scene structures. As a result, information about the same object in the scene has to be unnecessarily transmitted again and again. In block-matching, images are divided into square tiles, and simple translation is used to model object motions. However, square tiles rarely correspond to true object boundaries in the scene and simple translation can not model object motions very accurately. This leads to large residual error after motion compensation and results in inefficient coding. What is even worse is that the mismatch between the video representation and the scene structure makes it extremely difficult (if not impossible) for the user to access and manipulate the contents of video.

1.1.2 Object based multimedia

To solve the above problems, we should look toward video representation that are more physically and semantically related to scene structure. Instead of representing video as sequences of frames and breaking images into square blocks, we’d better describe video as collections of modeled objects that are encoded by computer vision algorithms and decoded according to scene description. In many cases, the outcome of the encoding looks just like a computer graphics database and script [3]. This concept will probably see its first widespread application in MPEG-4, which is the
1.1 Object based multimedia

first attempt to standardize an object based media representation. MPEG-4 is a new standard developed by MPEG (Moving Picture Experts Group), the committee that also developed the MPEG-1 and MPEG-2 video coding standards. A brief introduction to MPEG-4 is given below to illustrate the concept of object based media representation.

MPEG-4 audiovisual scenes are composed of a set of aural, visual or audiovisual contents, called “media objects” [4]. These media objects can be of natural or synthetic origin, either recorded with a camera or microphone, or generated with a computer. Still images (e.g. a fixed background), arbitrarily shaped video objects (e.g. a talking person - without the background) and audio objects (e.g. the voice associated with a person) are examples of primitive media objects. A media object in its coded form is represented independent of its surroundings or background, and consists of descriptive elements that allow handling the object in an audiovisual scene. Given a set of media objects and a scene description, an image sequence can be conveniently generated.

Figure 1.1 shows how an MPEG-4 scene is described as a set of individual objects. The figure contains compound media objects that group primitive media objects together. As an example: the visual object corresponding to the talking person and the corresponding voice are tied together to form a new compound media object, containing both the aural and visual components of that talking person.

Figure 1.2 depicts the basic concept for coding an MPEG-4 video sequence using a sprite panorama image. It is assumed that the foreground object (tennis player in the top right image) can be segmented from the background and that the sprite panorama image (the top left image) can be extracted from the sequence prior to coding. (A sprite panorama is a still image that describes the content of the background over all frames in a video sequence. Image mosaicing techniques [50, 51] can be used to construct a panorama image.) A frame in the original sequence (the bottom image) can be constructed by composing both the foreground and the background images according to scene description (e.g. camera parameters relevant to the background and motion parameters of the foreground).
Figure 1.1: An MPEG-4 scene
1.1 Object based multimedia

Figure 1.2: Sprite coding of MPEG-4 video sequence

The advantages of such an object based media representation are obvious. High compression can be achieved because the video representation is physically and semantically related to scene structures. Take Figure 1.2 for instance, in order to construct the original video sequence at the receiver, the large sprite panorama image is transmitted to the receiver only once and is stored in a sprite buffer. Only the foreground information (an arbitrarily shaped video object) and the scene description need to be updated over time. Apparently, this will result in high compression. What is more important is that the object based media representation enables the production and distribution of video contents that have far greater flexibility, functionality, interactivity and reusability than what can be obtained with today’s individual technologies such as digital television, animated graphics and the World Wide Web (WWW). In the case of MPEG-4, for example, users can construct complex scenes and manipulate meaningful objects in the scene through the following operations:

- Place media objects anywhere in a given coordinate system.

- Apply transforms to change the geometrical or acoustical appearance of a media
1.2 Object extraction and motion based segmentation

Object extraction and motion based segmentation

- Group primitive media objects to form compound media objects.
- Apply streamed data to media objects, in order to modify their attributes (e.g. a sound, a moving texture belonging to an object).
- Change, interactively, the user's viewing and listening points anywhere in the scene.

Apparently, object based video coding, such as MPEG-4, requires very complicated (and computationally expensive) image analysis algorithms. For example, in order to convert (encode) an ordinary video sequence into MPEG-4 format, image segmentation, motion estimation and temporal integration are needed to extract 2-D or 3-D motions and structures of different objects in the scene, and to establish the depth ordering and occlusion relationship between them. Compared to traditional video coding, object based video coding is also less robust because it sometimes fails to interpret the scene structure correctly. However, with the development of more intelligent algorithms and more powerful hardwares, these problems will eventually be solved, and we will see object based media representation in more and more applications.

1.2 Object extraction and motion based segmentation

The first (and probably the most important and difficult) step towards an object based video representation is object extraction. Different objects in the scene have to be identified and extracted from the video sequence. Basically, object extraction is a segmentation problem [5]. Unless object masks are available already, segmentation is needed in order to partition an image into a set of regions that correspond to different objects. However, object segmentation is an extremely difficult task. In fact,
it remains an unsolved problem in the areas of image processing and computer vision. Although object based video representation has been standardized in MPEG-4, method for object extraction is not specified in the standard, partly because good object segmentation algorithms are not available. In this work, we address the problem of object segmentation by means of motion based image sequence segmentation.

1.2.1 What is an object?

A fundamental question for object segmentation is: what is an “object”, or what is the homogeneous criterion used in object segmentation? There is no simple answer to this question. Different criteria could lead to totally different segmentation results for the same input data. Generally, three types of definitions for “object” are used in the literature:

- The first one is based on low-level features such as color or texture. An object is defined as a region with homogeneous color or texture. This is widely used in still image segmentation and region based image coding, but it usually results in over segmentation because one object may have different colors or textures.

- The second one is based on mid-level machine vision concepts such as coherent motion. An object is defined as a group of regions or pixels that undergo coherent motion. This is by far the most widely used definition for object segmentation. Depth is another useful mid-level concept for object extraction [6], in which case an object is defined as a group of surfaces with similar depth. But depth information is mainly used in the stereo literature and is difficult to extract from ordinary video sequences. (Although numerous algorithms for Structure From Motion computation have been proposed in the motion analysis literature, reliable recovery of depth information remains an open question.)

- The third definition is high level semantics. Human knowledge and experience is a critical factor for a semantics-based system. For example, we can effortlessly recognize a car, neither from its color nor from its motion, we simply know that
it is a car. This is based on our knowledge and previous experience. Today's computer does not have this kind of ability, so semantics-based (or knowledge-based) systems can only cope with very simple objects and requires human assistance.

In our algorithm, coherent motion is chosen as the criterion for object segmentation for the following reasons:

- Motion plays a very important role in human visual system. Moving objects and changing environment surround us. Even stationary objects appear to have relative motions because of our own motion or the movement of our eyes. Therefore, the presence of structures related to motion provides powerful cues for object segmentation.

- Motion information is required in the object based video representation as a major component of scene description.

- Motion has been studied in the fields of computer vision, image processing and psychology for a long time. Reliable method is available for motion estimation.

1.2.2 Problem definition and assumption

Our goal is to detect moving objects in the scene and segment them out. The basic assumption here is that pixels or regions undergoing similar motions are likely to arise from the same object in real world. The computer takes a sequence of video frames as input and generates a set of object masks along with their motion information. Each object mask consists of pixels that undergo coherent motion. Note that pixels belonging to one object mask may not necessarily be spatially connected because certain parts of an object may be occluded by other objects in the scene.

Instead of simple translation which is used in MPEG-2, we use more advanced parametric motion models, such as affine transform, to describe object motions. Therefore, pixels with different velocities may be considered as coming from one
object as long as their motions can be described by a single motion model. For example, pixels corresponding to different points on a rotating object move with different velocities, but their motions can be described by a single rotation, which is a special case of affine transform, so they will be grouped together as one object in our algorithm.

As color and texture, coherent motion is also an imperfect criterion for object segmentation because different parts of an object may move differently. For example, a walking person usually can not be segmented as a single object based on coherent motion, because the motions of the person’s arms, legs and body are different and can not be described by a single geometric transform. In this case, more sophisticated motion analysis and recognition techniques are needed [52]. Problem also arises when objects of interest do not move at all, in which case these objects will be grouped with the stationary background. It may ultimately be possible to build a fully automatic semantics-based image analysis system that represent objects in an image sequence with 3-D models by estimating 3-D motions, extracting 3-D structures and performing 3-D object recognition, but it will be many years before such techniques can be applied to arbitrary images.

1.2.3 Applications

Although the main motivation for this work is object based media representation, motion based object segmentation is also very useful for many other tasks such as image mosaicing, surveillance/target tracking, human-computer interaction and special effects in film making or commercial production. Briefly, in image mosaicing [50, 51], a panoramic view of the scene (usually the background) is obtained by aligning all the frames of an image sequence according to motion information and then using temporal integration to seamlessly cement the aligned frames into a spatially extended image. Motion segmentation is required to separate moving objects from the background before mosaic construction. In target tracking, moving targets have to be detected, segmented and tracked. In human-computer interaction, one of the goals is to make computers understand their users’ gestures [52]. Person segmenta-
tion and tracking have to be performed before gesture recognition. In special effects production, it is often necessary to segment out a moving object or change the way an object move. This also requires motion based segmentation.

1.3 Previous work

Motion based segmentation algorithms usually consist of two main components: (1) motion estimation; (2) identifying regions with coherent motion. Accurate motion estimation is essential for a good segmentation. But when a scene contains multiple moving objects (which is usually the case in real world video sequences), a good segmentation is needed for accurate motion estimation. The difficulty is that neither the correct segmentation nor the motion information are known a priori. This is a "chicken and egg" situation between segmentation and motion estimation.

Several techniques have been proposed to cope with this difficulty. Bergen et al. [7] introduced a two-component motion model that allows the analysis of many basic local motion configurations, including transparent motions, that do not conform to the traditional "single motion in a small neighborhood" model. In their algorithm, three consecutive frames were used to separate and estimate the two-component motions iteratively without knowing either motion a priori. Irani et al. [8, 9] proposed a method for detecting and tracking multiple moving objects one after another using dominant motion analysis. A single dominant motion is first detected, and the object corresponding to this motion is identified. This dominant object is then tracked over several frames using temporal integration. Once a dominant object has been detected and tracked, it is excluded from the region of analysis, and the process is repeated on the remaining region to find other objects and their motions. The above two methods are similar in that both assume that a dominant motion can be detected. Such methods may have difficulties with scenes containing several strong motions, since the estimated motion parameters reflect a mixture of different motions.

Adiv [10] proposed a two-stage technique that segment the optical flow field (motion field) into different regions corresponding to different moving objects. In the
first stage, a generalized Hough transform was used to partition the flow field into a set of segments, each of which contains connected flow vectors that are consistent with a rigid motion of a roughly planar surface. In the second stage, segments are grouped under the hypothesis that they are induced by a single rigidly moving object. Each hypothesis is tested by searching for the 3-D motion parameters which are compatible with all the segments in the corresponding group. This technique makes it possible to deal with independently moving objects by employing all the information associated with each object. However, optical flow field is noisy and partially incorrect, especially along motion boundaries. So segmentation based only on optical flow field can not give accurate result.

Derrell and Pentland [11] introduced a layered motion model to cope with multiple motions in the scene. Images are decomposed into several layers. Each layer contains a set of pixels that are moving coherently in the scene. They used parallel robust estimation technique and minimal-covering optimization to estimate the number of objects (layers). Simple translation models were used to describe their motions. Wang and Adelson [1,12,13] extended the layered representation to the case of an entire image sequence. They used least-squares technique to fit affine motion models to optical flow field. K-means clustering in the affine motion parameter space was used to group pixels into objects. The affine-fitting and K-means clustering processes iterate until convergence. After the segmentation masks for all the frames in a sequence were obtained, they used temporal integration to recover partially occluded regions of objects and determine the depth ordering for each object (layer). The above two methods are also based only on motion information and therefore, can not generate accurate object boundaries.

Black and Jepson [14] first segmented an image into regions using intensity information. For each resulting region, they fitted a parametric motion model to the optical flow field within that region. These regions can then be grouped into different objects according to the motion parameters. This method is effective because motion boundaries usually correspond to intensity edges which can be located precisely, and therefore, pixels in a region obtained from intensity segmentation are likely to come
from the same object and undergo coherent motion. Following this approach, several algorithms [15, 16] have been proposed. They used static information (intensity, color and texture) to perform the initial segmentation and then used motion information to group the resulting regions into objects by clustering in the motion parameter space. Because of the use of static segmentation, these algorithms can locate object boundaries more precisely than those based only on motion information. However, when the scene is complicated, static segmentation usually results in a large number of small regions. Motion estimation for all these regions is neither reliable nor efficient, and consequently, clustering in the motion parameter space becomes unreliable.

Another approach is to incorporate motion and static information into a statistical framework. Jepson and Black [17, 18] used a probabilistic mixture model to explicitly represent multiple motions in a small region. When multiple motions are present, the motion estimates within a region form distinct clusters. They employed a simple extension of the EM-algorithm to isolate these clusters, estimate their likelihood and detect outliers. This mixture model is applicable at both motion boundaries and in regions containing multiple transparent motions. But again, because it uses only motion information, it can not generate precise object boundaries. Etoh and Shirai [19], Chalom and Bove [20] used color, motion and pixel position to model an image as a n-dimensional jointly Gaussian feature space, and performed segmentation and motion estimation simultaneously by clustering in the feature space. Such a statistical framework is a promising approach to employ multiple cues in image sequences for object segmentation. However, the above methods use very complicated statistical models to perform clustering. When the dimension of the feature space is too high, the clustering process becomes unstable (e.g. sensitive to noise, initial condition or small change in parameter setting).

In addition to their own limitations, a common drawback of all the algorithms described above is that they use only two frames to perform segmentation, without exploring more temporal information which is available from an image sequence. This limits their ability to segment images containing objects with similar (but different) motions.
1.4 Contributions of the thesis

In this thesis, we have developed and implemented a novel automatic, multi-frame, region-feature based object segmentation algorithm which combines the advantages of feature based methods and gradient based (optical flow based) methods. It uses the first two frames to extract salient region features, which are then tracked over several frames. A few motion models are obtained by feature clustering. Region classification based on these motion models is performed to generate the final segmentation masks. It distinguishes itself from other methods by the following characteristics:

1. Unlike optical flow based motion segmentation techniques which are widely used in the literature, our method is based on the extraction and processing of region features.

2. Unlike most of the existing motion segmentation techniques which use only two frames of a sequence for the whole segmentation process, our proposed method explores temporal information over multiple frames.

3. Instead of clustering in the motion parameter space, our region grouping is based on region classification according to a small number of motion models.

Such a multi-frame, region-feature based object segmentation algorithm has several advantages.

Firstly, feature extraction reduces the vast amount of data in an image, without necessarily eliminating salient information. Indeed, Brady [21] noted that “not all information is created equal”, and that different locations in an image impose differing degrees of constraint on the motion parameters. Feature extraction also enables different motions in the scene to be estimated independently without interfering with each other. On the other hand, unlike most of the feature based motion analysis methods which use point features [22], the features used in our algorithm are regions obtained from static image segmentation. This allows us to use gradient based techniques to perform parametric motion estimation. Moreover, since region feature extraction and correspondence is essentially a matching process, it provides us with
a very good initial guess for the gradient based parametric motion estimation. This greatly reduces the computational complexity while increases the accuracy of the estimates. A combination of feature based methods and gradient based methods might also help to cope with complicated scenes, as suggested by Irani and Anandan [23].

Secondly, compared to the traditional two-frame formulation, the multi-frame formulation used in our algorithm can resolve certain ambiguities, and thus result in more coherent interpretation of the scene over time. For example, when an object moves slowly against a stationary background or when two objects move with slightly different velocities, it is very difficult to detect the relative motion from only two consecutive frames. As a result, the objects may not be segmented out using only two frames. However, after a number of frames (10 frames for example), the relative displacement between the two objects becomes apparent, and therefore, they can be segmented easily. Moreover, the use of multiple frames is potentially a way of combating noise for motion estimation, which is sensitive to image noise. It might also lead to the recognition of higher level movements, like walking or running, which consist of a complex and coordinated series of events that can not be understood by looking at only two frames.

Finally, our region grouping process is different from other techniques in the literature. In [15, 16], motion parameters were estimated for all the regions obtained from static segmentation, and then clustering based on these motion parameters (e.g. K-means clustering in the parameter space) was performed to group regions into objects. Such a clustering algorithm is unreliable and computationally expensive, because static segmentation usually results in a large number of small regions whose motion parameters are difficult to be estimated accurately. Consequently, segmentation obtained from this approach is not very accurate in general. Instead of clustering in the motion parameter space, we first obtain a few motion models, which can represent different motions in the scene, by region feature extraction and trajectory clustering. Region grouping is then performed by assigning each region to the motion model that best describes its motion. This region grouping process is simple, efficient and robust.
1.5 Thesis outline

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Region feature extraction, explained in Chapter 2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 2</td>
<td>Motion estimation and feature tracking, explained in Chapter 3.</td>
</tr>
<tr>
<td>Stage 3</td>
<td>Feature clustering and region classification, explained in Chapter 4.</td>
</tr>
</tbody>
</table>

*Table 1.1: Three stages for the proposed object segmentation algorithm*

The three main stages for the proposed object segmentation algorithm are summarized in Table 1.1. Chapter 2 explains the region feature extraction process. Chapter 3 first describes the gradient based parametric motion estimation technique, and then presents a simple yet efficient region tracking method. Chapter 4 shows how a few motion models can be obtained by feature clustering, and how to perform region classification using these motion models. Chapter 5 first presents simulation results on both synthetic and real image sequences, and then concludes this thesis with discussions on limitations of the proposed method and some future directions.
Chapter 2

Region feature extraction

2.1 Feature based motion analysis

2.1.1 Introduction

In feature based motion analysis, salient features have to be extracted, matched and tracked over time before the computation of scene structure and motion. There are mainly three types of features: point (or corner), edge and region.

Point features are distinctive image points that are accurately locatable and recur in successive images. Popular point features include local maxima of directional variance, junctions, line terminations and distinguished points along edge curves (e.g. maximum curvature points [24], zeros and discontinuities of curvature [25]). The trajectories derived from point features are very popular because their interpretation is obvious. However, extracting point features and establishing/maintaining the correspondence relationship over a sequence of frames is extremely difficult because noise often leads to false features and mismatch. Additional difficulties may arise when features disappear/reappear over time and when occlusion boundaries introduce false features into the field of view. Although many motion analysis algorithms use point features, most of them do not address the problem of feature extraction and correspondence, instead, they assume that features have been extracted and correspondence has been established. This is the major limitation of these algorithms.
2.1 Feature based motion analysis

Point features are mainly used in factory- or laboratory-type environments, where physical corners arise from man-made objects. In other type of scenes, salient point features might be rare. Moreover, how to extract point features from color images has not been well studied. This also limits the usefulness of point features in practice.

Edges are loci of one-dimensional change, located where the change in intensity is significant in one direction. They are generally detected by finding either maxima in the first image derivative or zero-crossings in the Laplacian of the Gaussian of the image. Their usefulness is limited by the "aperture problem": without assumptions about the nature of the motion, only the component of the motion perpendicular to the edge can be determined. Despite the advent of "snakes" [26], arbitrarily curving edges are difficult to describe and track, and simultaneous tracking of multiple open edge contours with automatic snake initialization still remains an open question.

Regions (or blobs) correspond to smooth surface patches, or loci of zero-dimensional change. Until recently, regions were neglected in favor of points or edges; reliable extraction of points or edges was difficult enough, and region features were even harder to extract, since minor difference between frames (due to motion or noise) can lead to very different segmentations in consecutive frames. Advances in the past few years, however, have shown that region features can be reliably extracted from image sequences and tracked over time, and they are increasingly being used in video processing and vision applications [27, 28, 29, 30].

2.1.2 The case for region features

In this work, we choose region features for the following reasons:

- Our goal is not the so called Structure From Motion computation [53], or SFM for short, (In SFM, the 3-D coordinates and motions of points on moving objects are recovered from a sequence of frames.) but to segment moving objects and estimate their motions in the 2-D image plane. Regions as features are better suited for this task than points, because points do not provide much information about object boundaries, while on the other hand, object boundaries
usually correspond to region boundaries obtained from color/intensity based segmentation.

- Recent progress in still image segmentation, such as the work of [31], makes region feature extraction feasible. Compared to points, regions possess more attributes, such as region size, shape and average color/intensity, which supply strong discrimination capability. These attributes can reduce the possibility of mismatch significantly, making feature correspondence very reliable. It is also easier to track regions than to track points because regions are less susceptible to occlusion and noise.

- The use of region features allows us to combine feature matching and gradient based parametric technique for motion estimation, resulting in accurate motion parameters.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Static segmentation on the first two frames of a video sequence.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Region grouping based on hue or intensity.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Feature correspondence based on region size.</td>
</tr>
</tbody>
</table>

*Table 2.1: Three steps for feature extraction*

Our region feature extraction process consists of three steps, as summarized in Table 2.1. They are explained in the following sections.

### 2.2 Static image segmentation

The region feature extraction process starts with static image segmentation on the first two frames of an image sequence. We use the “Edge Flow” algorithm [31] to perform static segmentation. Compared to other still image segmentation techniques in the literature, the Edge Flow algorithm has several advantages:

- It generates accurate object boundaries. We do not require that the static segmentation algorithm can partition an image into regions that exactly corre-
spond to different objects without over segmentation, but its ability to precisely detect object boundaries is essential to the final segmentation result.

- The Edge Flow method facilitates the integration of different image attributes such as intensity/color, texture and illusory discontinuities into a single framework for boundary detection. So it can cope with a wide variety of images.

- A user defined image scale $\sigma$ is the only significant control parameter that is needed by the algorithm. It can effectively control the behavior of the algorithm. In general, a small $\sigma$ (4, for example) results in fine segmentation while a large $\sigma$ (10, for example) results in coarse segmentation.

![Figure 2.1: Edge Flow segmentation on one frame of the MPEG Table Tennis sequence. Left: original frame; Right: segmented image, with region boundaries superimposed on the original frame.](image)

Figure 2.1 shows the Edge Flow segmentation result on one frame of the MPEG Table Tennis sequence. A small $\sigma$ is used in our implementation since our goal in this step is to detect object boundaries as accurately as possible. It can be seen that most of the object boundaries are accurately detected, although the result contains a large number of (highly oversegmented) small regions. *The problem of how to group these small regions into meaningful objects is addressed in the rest of the thesis.*

The main drawback of the Edge Flow algorithm is its computational complexity. It requires a large amount of memory and computation time. In fact, the Edge Flow
segmentation takes more than 50% of the computation time used in the whole motion segmentation process.

We do not give a detailed description of the Edge Flow algorithm in this thesis because it does not provide much insight into the overall object segmentation algorithm. Readers are referred to [31] for a detailed mathematical formulation.

It should be pointed out that the Edge Flow algorithm is just one of many possibilities for static segmentation. Any other method that generates accurate object boundaries could be employed, and the better the static segmentation, the better the final object segmentation result will be.

2.3 Region grouping

In the Edge Flow segmentation, the use of a small σ results in accurate object boundary detection. However, a problem arises: small spatial variation of intensity due to noise or different lighting conditions (termed shading in computer vision) is also detected. This results in significant oversegmentation. Even surface of an object with uniform color is segmented into a set of small regions.

To solve this problem, a simple hue-based region grouping is performed immediately after the Edge Flow segmentation. Adjacent regions with similar hue values are merged. Hue, $H$, is a color attribute used in the HSI color model. It describes a pure color of light (determined by the wavelength in the visible spectrum), such as yellow, orange or red. Saturation, $S$, is another color attribute that gives a measure of the degree to which a pure color is diluted by white light. Intensity, $I$, is the third component in the HSI color model, which describes the brightness of light. While saturation and intensity are affected by change in brightness due to noise and shading, hue is invariant to this change. Therefore, region merging based on hue can effectively eliminate the influence of noise and shading in static segmentation, and thus reduces the total number of regions significantly.

For a given pixel in a color image (in RGB color space), its hue value $H$ is
2.4 Region feature correspondence

The average hue of a region is defined as the arithmetic mean of the hue values of the pixels belonging to that region. In our implementation, \( T_h \) is set to \( 10^\circ \). Note that hue values within the intervals of \( (0^\circ, 10^\circ) \) and \( (350^\circ, 360^\circ) \) need special treatment in the implementation because they represent similar colors. Also, hue is not defined for pixels whose saturation is equal to zero (a pixel’s saturation is zero when it has the same \( R, G \) and \( B \) values). These pixels are excluded from hue computation.

The above hue-based region grouping is for color images. For grayscale images, the grouping is based on intensity value. Two adjacent regions are merged if the difference between their average intensity values is less than a threshold (say 5). The average intensity of a region is defined as the arithmetic mean of the intensity values of the pixels belonging to that region.

2.4 Region feature correspondence

After static segmentation on the first two frames of a video sequence, a set of region features are extracted from the resulting regions by finding correspondence between regions of the two consecutive frames. That is, for a given region in the first frame, if we can find a region in the second frame so that these two regions correspond to exactly the same surface (or object) in the scene, they are considered as a pair of corresponding region features.
2.4 Region feature correspondence

2.4.1 Introduction

The feature correspondence problem can be defined as: given 2 frames taken at different time instants or different view points, containing \( m \) and \( n \) primitives (points, edges or regions) respectively, match a primitive in the first frame to another primitive in the second frame. This is a combinatorially explosive problem, whose worst case enumeration is \( mn \) possible matches. The occurrence of occlusion and disocclusion also adds to the difficulty of the problem. Constraints from the physical world are usually imposed to reduce the computational effort. Popular constraints include:

- Unique match, which forces each primitive to match with only one other primitive.

- Maximum velocity, which implies that if the bound on velocity is known \( a \) \textit{priori}, one can limit the search for possible match in the next frame to a small neighborhood of the position of a given primitive in the present frame.

- Measures of feature similarity, which states that each feature has a set of attributes which are invariant under the appropriate transformation.

- Small velocity change, which assumes that the direction and speed of motion can not change by a large amount.

The first three constraints, namely \textit{unique match}, \textit{maximum velocity} and \textit{measures of feature similarity}, are used in our feature correspondence algorithm. The \textit{small velocity change} constraint is not used in the matching process, instead, it is used in the region tracking algorithm which is described in Chapter 3.

2.4.2 Feature correspondence based on region size

When regions are used as matching primitives, several region attributes such as size, shape and average color/intensity can be employed to verify matches and resolve ambiguities. In addition to the above attributes, relational structure (adjacency relation) of regions is also widely used in region correspondence [32, 33]. In stereo
vision and 3-D object recognition, the size and shape of a region may change significantly due to large change in view point, so color/intensity information and relational structure of regions have to be used in region matching. In our case, however, the small inter-frame motion (as a result of high frame rate) leaves the size and shape of a region largely unchanged between successive frames. Therefore, region size and shape are strong indicators for the similarity between regions in two successive frames, and are sufficient for region correspondence. In digital images, region size is simply the number of pixels in a region, and is very easy to calculate. The shape of a region is much more difficult to describe, so we do not use it explicitly in the correspondence process, instead, we use it implicitly to reject miscorrespondence.

In our implementation, a pair of regions in the first and the second frames are considered as a pair of corresponding features when they satisfy the following two requirements:

- The distance between their centroids is less than the expected maximum displacement of an object between two successive frames.
- Their sizes are similar (e.g. the difference between them is less than, say 2%, of the size of the smaller region).

Here, two parameters need to be set by the user. The first one is $D_m$, the expected maximum displacement (measured in number of pixels) of an object between two successive frames. It controls the size of the search window for correspondence. For region feature correspondence, it does not affect the computational complexity significantly because the number of regions in the search window does not increase dramatically even when the size of the window increases by a large amount. So $D_m$ can be set to a large value (say 15 pixels, which is large enough for most applications). This leads to a circular search window of 30 pixels in diameter. However, a large search window will introduce ambiguity when similar patterns repeatedly appear in the scene (the brick pattern on the wall is such an example), in which case we have to reduce the window size to avoid ambiguity.

The second parameter is $T_s$, the threshold for the (maximum) difference between
2.4 Region feature correspondence

the sizes of two regions. It controls the number of regions that can be extracted as features, and is a measure of confidence for a possible match. The smaller the threshold, the less the possibility of miscorrespondence. \( T_s \) is also an indicator of how well the displacement between the centroids of a pair of corresponding region features in two successive frames approximates the true displacement between these two corresponding features. A small \( T_s \) increases the accuracy of this approximation, which is used as an initial guess in the following motion estimation algorithm. Therefore, in order to reduce the chance of miscorrespondence and increase the accuracy of initial guess for motion estimation, we use a small threshold in our implementation. However, the sizes of a pair of corresponding region features in two frames can not be exactly the same due to the imperfection of static segmentation or small change in view point as a result of motion. So \( T_s \) should not be too small, otherwise, no features can be extracted, making the following analysis impossible.

![Figure 2.2: Region feature extraction for the Table Tennis sequence. Four features, coming from the ball, the racket, the table and the wrist respectively, are extracted. Left: the extracted features in frame 1; Right: the corresponding features in frame 2.](image)

Figure 2.2 shows the feature extraction result for the Table Tennis sequence. A set of salient features are extracted from different objects in the scene. Note that these region features are only a small portion of the regions obtained from static segmentation. This greatly reduces the amount of data that has to be processed in the following steps. In general, partially occluded regions will not be extracted as
features because their sizes change significantly between two frames due to occlusion. This makes the following motion estimation very reliable.

### 2.4.3 Rejecting miscorrespondence

The chance of miscorrespondence can be effectively reduced by using a small threshold $T_s$. Even miscorrespondence does occur, it can be rejected using regions’ shape information implicitly through a simple procedure described below.

**Miscorrespondence rejection based on region shape** After motion estimation for the extracted region features (which will be explained in the next Chapter), we align every pair of corresponding region features (warp one region in the first frame towards its corresponding region in the second frame) according to the estimated motion parameters. Suppose that miscorrespondence has occurred, that is, two regions coming from two different surfaces in the scene have been extracted as a pair of corresponding features. In such a case, their sizes must be similar since they satisfy the two requirements for region correspondence. However, their shape will not be the same because they arise from different surfaces in the scene. (This assumption may be violated when similar patterns repeatedly appear in the scene, in which case computationally expensive correlation-based region matching has to be employed to verify correspondence.) Therefore, the overlapping area of the two regions after motion alignment is much smaller than their actual sizes. This information leads to easy detection of miscorrespondence without examining the actual shape of a region.
Chapter 3

Motion estimation and feature tracking

3.1 Optical flow computation

After a set of region features are extracted from the first two frames of a sequence, motion estimation is needed to estimate their motions. There are essentially two approaches for this task. One is region matching, the other is optical flow computation.

In region matching [38], the translational motion of an arbitrarily shaped region between two frames is estimated by searching the vector (in 2-D image plane) which maximizes the correlation (or minimizes the sum of squared difference) of that region between two frames. Block matching is a special case of region matching, in which square blocks are used. It is widely used in traditional video coding schemes for its simplicity and regularity. Region matching can be considered, to a great extent, as a feature matching process. Actually, in many point feature matching algorithms, the correlation of a small region centered on the feature point is used as the confidence measure for a possible match. Region matching is not suitable for motion based segmentation because simple translation can not model many real world motions, such as rotation and scaling. However, given a large enough search space and small enough search step, it usually generates an accurate translation estimate, which can be used as a reasonable initial guess for optical flow based motion estimation.
3.1 Optical flow computation

Optical flow based approaches calculate the apparent velocities of brightness patterns in an image [39]. In general, the computation of optical flow is difficult and sensitive to noise. However, it only requires local image measurements and therefore has the advantage of being able to cope with complex and varying motion fields. So nearly all the motion based segmentation algorithms (including our proposed algorithm) require the computation of optical flow.

3.1.1 Constant brightness constraint equation

Constant brightness constraint equation

Optical flow estimation is based on the assumption of constant brightness: the brightness of a particular point in a brightness pattern is constant. This leads to the well known constant brightness constraint equation [39] that relates the change in image brightness to the motion of the brightness pattern

\[
\frac{\partial I(x, y, t)}{\partial x} \frac{dx}{dt} + \frac{\partial I(x, y, t)}{\partial y} \frac{dy}{dt} + \frac{\partial I(x, y, t)}{\partial t} = 0
\]  

(3.1)

Derivation of the constant brightness constraint equation Consider a small patch of a brightness pattern \( I(x, y, t) \) that is displayed at position \((x, y)\) and time instant \(t\). After a time interval \(dt\), the patch is displayed at position \((x + dx, y + dy)\) due to motion. The brightness of the patch is assumed to remain constant so that

\[
I(x, y, t) = I(x + dx, y + dy, t + dt)
\]

(3.2)

Expanding the right-hand side about the point \((x, y, t)\) we get

\[
I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I(x, y, t)}{\partial x} dx + \frac{\partial I(x, y, t)}{\partial y} dy + \frac{\partial I(x, y, t)}{\partial t} dt + \epsilon
\]

(3.3)

where \(\epsilon\) contains second and higher order terms in \(dx\), \(dy\) and \(dt\). After subtracting \(I(x, y, t)\) from both sides and dividing through by \(dt\) we have

\[
\frac{\partial I(x, y, t)}{\partial x} \frac{dx}{dt} + \frac{\partial I(x, y, t)}{\partial y} \frac{dy}{dt} + \frac{\partial I(x, y, t)}{\partial t} + O(dt) = 0
\]

(3.4)
3.1 Optical flow computation

where $O(dt)$ is a term of order $dt$ (we assume that $dx$ and $dy$ vary as $dt$). In the limit as $dt \to 0$, this becomes

$$\frac{\partial I(x,y,t)}{\partial x} \frac{dx}{dt} + \frac{\partial I(x,y,t)}{\partial y} \frac{dy}{dt} + \frac{\partial I(x,y,t)}{\partial t} = 0$$  \hspace{1cm} (3.5)$$

The velocity vector $(u(x,y) = dx/dt, \ v(x,y) = dy/dt)^T$ is what we solve for and is called optical flow vector. The optical flow vector is not exactly equal to the true motion, but it is a good approximation when the first order Taylor series approximation of $I(x,y,t)$ holds

$$I(x + dx, y + dy, t + dt) \approx I(x,y,t) + \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt$$  \hspace{1cm} (3.6)$$

Least-squares minimization

The optical flow of an image point cannot be computed independently of the neighboring points without introducing additional constraints, because the velocity at each image point has two components while the constant brightness constraint equation provides only one constraint. Smoothness of Motion was introduced by Horn and Schunck [39] as an additional constraint, which states that neighboring points have similar velocities and the velocity field varies smoothly almost everywhere. In another word, it assumes that all the pixels in a sufficiently small region undergo the same translation. One way to express this constraint is to use least-squares minimization. First we rewrite Equation 3.1 as

$$-I_t \approx I_x u(x,y) + I_y v(x,y)$$  \hspace{1cm} (3.7)$$

where $I_x$, $I_y$ and $I_t$ are the partial derivatives of the image intensity $I$ at position $(x,y)$ with respect to $x$, $y$ and $t$. Define the error over a small neighborhood $r$ as

$$E = \sum_r e^2 = \sum_r (I_t + I_x u(x,y) + I_y v(x,y))^2$$  \hspace{1cm} (3.8)$$

where $e = I_t + I_x u(x,y) + I_y v(x,y)$. Minimizing $E$ with respect to $u(x,y)$ and $v(x,y)$ yields:

$$
\begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix}
\begin{bmatrix}
u(x,y) \\
v(x,y)
\end{bmatrix}
= -
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
$$  \hspace{1cm} (3.9)$$
The summation is applied to the neighborhood $r$, which is typically a $3 \times 3$ or $5 \times 5$ window centered at $(x, y)$. This amounts to assuming implicitly that the flow field varies smoothly over the image.

Equations 3.8 and 3.9 can be expressed in vector form as

$$E = \sum_r (\Delta I + \nabla I \cdot \mathbf{U}(x))^2$$

$$\left[ \sum_r (\nabla I)^T(\nabla I) \right] \mathbf{U}(x) = - \sum_r (\nabla I)^T \Delta I$$

where $x = (x, y)$, $\mathbf{U}(x) = (u(x), v(x))^T$, $\nabla I = (I_x, I_y)$, which is the spatial gradient, and $\Delta I = I_t = I(x, t) - I(x, t - 1)$, which is the difference between the intensity values of two successive frames of an image sequence (or the partial derivative of intensity $I$ with respect to $t$).

Iterative solution

The above algorithm is usually implemented in an iterative fashion. Assume that we have an approximate flow field from previous iteration $\mathbf{U}_{i-1} = (u_{i-1}, v_{i-1})^T$. The incremental flow vector $\delta \mathbf{U} = (\delta u, \delta v)^T$ can be obtained by minimizing the error measure

$$E(\delta \mathbf{U}) = \sum_r (\Delta I + \nabla I \cdot \delta \mathbf{U}(x))^2$$

where $\Delta I = I(x, t) - I(x - \mathbf{U}_{i-1}(x)^T, t - 1)$, which is the difference between the intensity values of two successive frames, after taking the previous estimates into account.

The solution is given by

$$\left[ \sum_r (\nabla I)^T(\nabla I) \right] \delta \mathbf{U} = - \sum_r (\nabla I)^T \Delta I$$

The current estimate is obtained as

$$\mathbf{U}_i = \mathbf{U}_{i-1} + \delta \mathbf{U}$$

convergence is reached when the motion estimate $\mathbf{U}$ remains unchanged, or equivalently, when $\delta \mathbf{U}$ goes to zero.
3.1 Optical flow computation

3.1.2 Coarse-to-fine strategy

The optical flow computation is accurate for small motions, but it will fail when the image motion is large, because Equation 3.6 no longer holds. Region matching can cope with large motion, but a matching process that must accommodate large displacement is very expensive in computation. To cope with large motion, Lucas and Kanade [42] suggested a simple yet effective multi-resolution coarse-to-fine motion estimation strategy. It is based on the fact that large motion in high resolution images becomes small in low resolution images. The coarse-to-fine strategy begins with constructing an image pyramid, such as the Gaussian pyramid proposed by Burt and Adelson [40], which contains a set of images with different resolutions. Low resolution image is obtained by smoothing (low pass filtering) and downsampling high resolution image. A coarse motion field is first estimated from the low resolution, smoothed version of the original images. It is propagated via a pyramid expansion operation as described in [40] to the higher resolution level, where it is used to warp higher resolution images to roughly stabilizing moving objects. Finer incremental motion estimates are then obtained using higher resolution images. Again, this is implemented in an iterative fashion.

The multi-resolution coarse-to-fine strategy is also necessary for solving the temporal aliasing problem, which occurs when the temporal sampling rate (frame rate) of an image sequence is lower than that required by the sampling theorem to uniquely represent high spatial frequency components undergoing large motion. The classic example of temporal aliasing is the wagon wheel effect in old western movies: when the wheel rotates at the right speed relative to the frame rate, its direction of rotation appears to reverse. Temporal aliasing is the source of false matches in correspondence solutions or (equivalently) local minima of the objective function used for minimization, which is a common problem in computing image velocities in real video sequences. An important observation, made by Lucas and Kanade [42], concerning this type of aliasing is that it only affects the high spatial frequencies of an image. Therefore, the multi-resolution coarse-to-fine strategy can reduce the temporal aliasing effect.
In addition to its ability to cope with large motion and temporal aliasing, the coarse-to-fine strategy also reduces the computational load and thus speeds up the motion estimation process significantly, because a large portion of the estimation is done using low resolution images which have much fewer pixels than the original images.

### 3.1.3 Limitations

There are several problems that limit the usefulness of optical flow computation. Firstly, it is susceptible to the aperture problem, which, in many cases, only allows the precise computation of the normal flow, i.e. the velocity component parallel to the image gradient. This problem is a direct result of the local nature of the optical flow computation. Verri and Poggio [41] showed that in general, the optical flow field differs from the true image motion field, except where image gradients are strong, e.g. at edge or corner locations.

Secondly, optical flow computation is prone to boundary oversmoothing, i.e. the estimated motion field around object boundaries is a mixture of true motions of different objects. This is a direct result of the *Smoothness of Motion* constraint used in the optical flow estimation.

Thirdly, the gradient based method described above computes the best estimate of translational flow in each local neighborhood of an image. Theoretically, there is an elegant set of equations relating motions to image brightness, but the solution of these equations is not well behaved numerically and is highly sensitive to noise, because the computation depends on taking derivatives, which is an inherently noise augmenting process.

Different strategies have been proposed to address these problems. For example, global region matching can effectively resolve the aperture problem; static image segmentation is performed prior to motion estimation to avoid oversmoothing; parametric motion models are introduced to suppress noise and to cope with complex motions. In the next section, we show how to combine these strategies to perform
accurate motion estimation.

3.2 Region based parametric motion estimation

The optical flow computation described earlier requires least-squares minimization in a small neighborhood, such as a $3 \times 3$ or $5 \times 5$ window. A small window provides very few constraints on the least-squares calculation and thus makes it sensitive to noise. Increasing the window size can reduce the impact of noise, but the assumption that all the pixels within the window undergo the same translational motion is more likely to be violated, and outliers will be introduced. To overcome this problem, parametric model based approaches have been proposed to estimate more complex motions over much larger image neighborhoods. In these approaches, it is assumed that image motion within a region (possibly the entire image) can be modeled by a geometric transformation, such as affine or projective transformation, which is described by a few parameters. The velocity of each pixel can be easily calculated using these parameters.

There are two reasons which make parametric model based approaches effective. Firstly, geometric transformations are accurate enough to model many real world motions. Although it is impossible to describe deformable motion (such as human facial expression) using a single geometric transform, it is still a reasonable approximation of the motion field in a local neighborhood [61]. Secondly, the small number of model parameters (e.g. 6 in the case of affine transformation) provide a concise description of complex image motions over a large region. As a result, motion estimation can be very accurate since only a few motion parameters are estimated from a huge number of constraints.

3.2.1 Geometric transformation

Consider an intensity pattern $I(x_1, y_1, t)$, which is displayed at $(x_1, y_1)$ in frame $F_t$. Suppose that this pattern moves to $(x_2, y_2)$ in frame $F_{t+1}$ due to motion, and becomes $I(x_2, y_2, t + 1)$. In parametric motion estimation, the two sets of coordinates, $(x_2, y_2)$
and \((x_1, y_1)\), are related by a geometric transformation function. Several widely used transformations are shown in Figure 3.1 and are described below.

- **Rigid-body transformation**

  \[
  \begin{bmatrix}
  x_2 \\
  y_2
  \end{bmatrix} =
  s \begin{bmatrix}
  \cos \theta & -\sin \theta \\
  \sin \theta & \cos \theta
  \end{bmatrix} \begin{bmatrix}
  x_1 \\
  y_1
  \end{bmatrix} +
  \begin{bmatrix}
  x_0 \\
  y_0
  \end{bmatrix}
  \]

  (3.15)

  It has four parameters: scale \(s\), rotation \(\theta\), and translation \(x_0\) and \(y_0\). Object under rigid-body transformation retains their relative size and shape.

- **Affine transformation**

  \[
  \begin{align*}
  x_2 &= a_1 + a_2 x_1 + a_3 y_1 \\
  y_2 &= a_4 + a_5 x_1 + a_6 y_1
  \end{align*}
  \]

  (3.16)

  It has 6 parameters and is a general case of rigid-body transformation. It can model vertical and horizontal shear, and preserves parallel lines and equispaced points. Note that the inverse of an affine (or rigid-body) transformation is also an affine (or rigid-body) transformation.

- **Projective transformation**

  \[
  \begin{align*}
  x_2 &= \frac{a_{11} x_1 + a_{21} y_1 + a_{31}}{a_{13} x_1 + a_{23} y_1 + a_{33}} \\
  y_2 &= \frac{a_{12} x_1 + a_{22} y_1 + a_{32}}{a_{13} x_1 + a_{23} y_1 + a_{33}}
  \end{align*}
  \]

  (3.17)

  It has 8 independent parameters and preserves lines in all directions. However, parallel lines in the original image are no longer parallel after projective transformation, instead, they converge to a vanishing point. This property, called perspective effect, which characterizes the image formation process of cameras and human eyes, is the basis for many 3-D motion analysis and scene modeling algorithms.

- **Bilinear transformation**

  \[
  \begin{align*}
  x_2 &= a_0 x_1 + a_1 y_1 + a_2 x_1 y_1 + a_3 \\
  y_2 &= a_4 x_1 + a_5 y_1 + a_6 x_1 y_1 + a_7
  \end{align*}
  \]

  (3.18)
3.2 Region based parametric motion estimation

It has 8 parameters and preserves lines that are horizontal or vertical in the original image. Lines that are not oriented along these two directions in the original image become curves in the transformed image.

Figure 3.1: Examples of geometric transformation. Top left: original image pattern; Top middle & Top right: image patterns after affine transformations; Bottom left: rigid-body transformation; Bottom middle: projective transformation; Bottom right: bilinear transformation.

3.2.2 Affine motion model

Among these transformations, affine model is the most widely used in the literature. We also use it in our implementation for its ability to model the 2-D projections of complex motions in 3-D space under parallel projection. For example, 3-D rotation, when projected onto the image plane, results in a 2-D affine transformation. Another advantage of affine model is its simplicity. Motion estimation using affine model only involves linear minimization. Projective and bilinear transformations can model more complex motions, but estimating their parameters requires nonlinear optimization,
which is computationally expensive and numerically unstable. Since our purpose is to estimate 2-D motions of the extracted region features, affine models suffice.

Equation 3.16 relates \((x_2, y_2)\) and \((x_1, y_1)\) by an affine transformation. Equivalently, the affine model of image velocity at \((x_1, y_1)\) is given by (assume \(dt = 1\)):

\[
\begin{align*}
    u(x_1, y_1) &= x_2 - x_1 = a_1 + (a_2 - 1)x_1 + a_3y_1 \\
    v(x_1, y_1) &= y_2 - y_1 = a_4 + a_5x_1 + (a_6 - 1)y_1
\end{align*}
\] (3.19)

therefore, the relationship between affine transformation and affine motion model is very simple. Given one, the other can be easily derived. For clarity, we will use the following notation to represent affine motion:

\[
\begin{align*}
    u(x, y) &= a_{x0} + a_{xx}x + a_{xy}y \\
    v(x, y) &= a_{y0} + a_{yx}x + a_{yy}y
\end{align*}
\] (3.20)

This is also an affine transformation. As mentioned earlier, an advantage of the affine motion model is its ability to describe complex motions with only 6 parameters.

### 3.2.3 Affine motion estimation

There are two techniques for affine motion estimation. The first one is "affine fitting", which was proposed by Negahdaripour and Lee [43], and was later used by Wang and Adelson [12] in their motion segmentation algorithm. The second one is "affine flow", which was proposed by Bergen et al. [44].

#### Affine fitting

In "affine fitting", optical flow field is first estimated using Horn and Schunck's algorithm [39]. Then an affine motion model is fitted to the optical flow field of a particular region \(r\) using least-squares minimization. Rewrite affine motion equation 3.20 in vector form as

\[
U(x) = Xa
\] (3.21)

where \(U(x) = (u(x, y), v(x, y))^T\), \(x = (x, y)\), \(a = [a_{x0}, a_{xx}, a_{xy}, a_{y0}, a_{yx}, a_{yy}]^T\) and

\[
X = \begin{bmatrix}
    1 & x & y & 0 & 0 & 0 \\
    0 & 0 & 0 & 1 & x & y
\end{bmatrix}
\]
Suppose that the optical flow field within region \( r \) can be described by an affine model, and define an error term as

\[
E = \sum_r (U(x) - Xa)^T (U(x) - Xa)
\]  

(3.22)

Minimizing \( E \) with respect to \( a \), we get

\[
\left[ \sum_r X^T X \right] a = \sum_r X^T U(x)
\]  

(3.23)

therefore

\[
a = \left[ \sum_r X^T X \right]^{-1} \sum_r X^T U(x)
\]  

(3.24)

where \( r \) is the region of interest, and the summation is applied to this region.

**Affine flow**

Alternatively, Bergen et al. [44] proposed the “affine flow” approach, which directly incorporate affine models into the constant brightness constraint equation. Insert 3.21 into equation 3.10, we get

\[
E = \sum_r (\Delta I + \nabla I \cdot U(x))^2 = \sum_r (\Delta I + \nabla I \cdot Xa)^2
\]  

(3.25)

Minimizing \( E \) with respect to \( a \) leads to

\[
\left[ \sum_r X^T (\nabla I)^T (\nabla I) X \right] a = -\sum_r X^T (\nabla I)^T \Delta I
\]  

(3.26)

and therefore

\[
a = \left[ \sum_r X^T (\nabla I)^T (\nabla I) X \right]^{-1} \left[ -\sum_r X^T (\nabla I)^T \Delta I \right]
\]  

(3.27)

where \( \nabla I = (I_x, I_y) \), \( \Delta I = I(x,y,t) - I(x,y,t-1) \) and \( r \) is the region of interest within which the optical flow field can be modeled by affine motion. Like direct optical flow computation, the “affine flow” algorithm can also be implemented in an iterative fashion with a multi-resolution coarse-to-fine strategy.

The two techniques described above are similar in that both use least-squares minimization. In practice, however, the “affine flow” approach performs better than “affine fitting”. Steve Mann [45] discussed this in detail and he further extended the
two techniques to the 8-parameter projective transformation case. In short, “affine fitting” tries to fit affine model to optical flow field, whose computation relies on the assumption that pixels in a small neighborhood move with the same translational motion. Paradoxically, affine model forces the flow field to vary everywhere. This is equivalent to fitting a model to a data set that actually can not be described very well by that model. Consequently, the resulting model parameters are not very accurate. In contrast, the “affine flow” method incorporates affine model directly into the constant brightness constraint equation. The obvious benefit of this approach is that the resulting flow field is more consistent with the motion model, and therefore, the estimated model parameters are more accurate. For this reason, we choose “affine flow” as our affine parameter estimator.

3.2.4 Limitations

Affine model enforces strong constraints on the spatial variation of the image motion within a region. When the model is a good approximation of the motion field, this method is very accurate since one only has to estimate a small number of parameters given hundreds of thousands of constraints. However, affine motion estimation relies on the assumption that the motion within a region can be described by an affine model. This assumption will be violated when a scene contains multiple motions which can not be described by a single model. In this case, we have to partition an image into a set of regions, each containing only one motion, and then apply affine motion estimation to each of these regions. So good segmentation is needed for accurate motion estimation. But unfortunately, image segmentation is exactly what we want to achieve, and accurate motion estimation is essential for good segmentation. The difficulty is that neither the correct segmentation nor the motion information are known a priori. This is a “chicken and egg” situation between segmentation and motion estimation. To overcome this problem, researchers have proposed to segment images using color/intensity information, and then apply affine motion estimation to the resulting regions [14, 15, 16]. This approach is effective because intensity edges usually correspond to motion boundaries, and therefore, pixels within a region re-
3.2 Region based parametric motion estimation

resulted from color/intensity segmentation generally undergo coherent motion. When
the scene is simple, good static segmentation can be achieved, making motion estimation easy and accurate. However, when the scene is complicated, static segmentation usually results in a large number of small regions. Motion estimation for these regions is difficult because:

- They present strong homogeneity in color/brightness, which leads to the aperture problem, i.e. motion information can only be accurately recovered at locations where image gradients are strong.

- Their sizes are usually small. A small number of pixels in a region may not impose enough constraints on least-squares minimization, making it sensitive to noise and outliers.

- To cope with large motion, multi-resolution coarse-to-fine strategy is needed. However, small region has little or no support map at the low resolution level, making the coarse-to-fine strategy infeasible.

To illustrate these problems, let us consider the example shown in Figure 3.2. The first row in the figure shows two consecutive frames of a synthetic image sequence. There are two bars in the scene, both consist of five regions with different gray scales (the gray scale within each region is uniform). One bar is moving to the right, occluding part of the other one which is moving to the left. The true motions in this example are pure horizontal translations of (3,0) and (-3,0) pixels per frame for the two bars respectively. Due to the aperture problem, however, we can not determine whether the bars are moving horizontally or vertically if we observe the scene within a small window. This problem is clearly seen from the horizontal and vertical components of the motion field\(^1\) shown in the second row of Figure 3.2, which is obtained using direct optical flow estimation. Only pixels along intensity edges have nonzero velocities, but the velocities contain equally strong vertical and horizontal components. If we first segment the image into a set of regions using

\(^{1}\)The motion field is heavily quantized for display.
Figure 3.2: The synthetic Two-bars sequence (see text). First row: two consecutive frames; Second row: horizontal and vertical components of the motion field estimated using direct optical flow estimation; Third row: horizontal and vertical components of the motion field estimated using region based affine motion estimation. In the motion fields, different velocities are depicted with different gray scales, with darkness proportional to the leftward and upward velocities.
intensity information, and then apply region based affine motion estimation to these regions, we can get better result when the motion is small. When the motion is large, however, we have to use the coarse-to-fine strategy, but small regions might disappear at the coarse level, making motion estimation impossible. The bottom row in Figure 3.2 shows the horizontal and vertical flow fields obtained by affine motion estimation, which are much better than those estimated using direct optical flow computation, but they still have large errors in many regions (especially in partially occluded regions).

3.3 Motion estimation with initial guess

3.3.1 Initial guess

Like [14, 15, 16], we also apply parametric motion estimation to regions obtained from static segmentation. The difficulties mentioned in the previous section are overcome by our algorithm because we perform motion estimation only for the extracted region features, and good initial guess for their motion parameters is available.

In our implementation, the displacement between the centroids of a feature's masks in the first two frames is used as the initial guess for motion estimation (recall that for each feature, we have its region masks in the first two frames). The position \((x_c, y_c)\) of the centroid of a region is calculated as follows

\[
x_c = \sum_r x / N , \quad y_c = \sum_r y / N
\]

where \(r\) is the region of interest, \(N\) is the number of pixels in the region. This initial guess is not an arbitrary choice. It is a very accurate approximation of the translational component of the feature’s motion. Classical mechanics dictates that with respect to an external force, the velocity of a rigid body can be represented by the motion of its centroid. Although the centroid's displacement does not provide information about the rotational component of a region's motion, it is very effective as initial guess in case of large motion, because in most cases it is large translation that makes motion estimation difficult. If large rotational motion is present in the scene,
3.3 Motion estimation with initial guess

a computationally expensive search for the rotation parameter has to be performed in order to obtain an initial guess.

Compared to direct optical flow estimation, in which case only a small number of pixels is used for the computation of motion vector at each location, the region feature extraction process (see Chapter 2) is based upon the attributes of the whole region, it is in essence a global region matching process. Therefore, the ambiguity resulted from the aperture problem can be effectively resolved by using the displacement of region features as initial guess for motion estimation.

3.3.2 Implementation of the "affine flow" technique

In our algorithm, affine motion parameters for each region feature are estimated using the "affine flow" technique, which is implemented in an iterative fashion. Because good initial guess is available, the multi-resolution coarse-to-fine strategy is not required. There are two basic components in the computation:

1. Affine parameter estimation.

2. Image warping.

They are explained as follows:

**Affine parameter estimation**

Let \( \mathbf{a}_{i-1} \) denote the parameters estimated from the previous iteration. We calculate an approximate flow field \( \mathbf{U}_{i-1} \) using \( \mathbf{a}_{i-1} \) according to Equation 3.21. From Equation 3.27, a set of incremental affine parameters \( \mathbf{\delta a} \) can be obtained as

\[
\mathbf{\delta a} = \left[ \sum_r \mathbf{X}^T(\nabla I)^T \nabla I \right]^{-1} \left[ -\sum_r \mathbf{X}^T(\nabla I)^T \Delta I \right] 
\]

where \( r \) is the region feature of interest, \( \nabla I = (I_x, I_y) \) is the spatial gradient, and

\[
\Delta I = I_t = I(\mathbf{x}, t) - I(\mathbf{x} - \mathbf{U}_{i-1}(\mathbf{x})^T, t - 1)
\]

which is the difference between the intensity values of two successive frames, after taking the previous motion estimates into account. The affine parameters \( \mathbf{a}_i \) for the
current iteration are given by
\[ a_i = a_{i-1} + \delta a \] (3.31)

\( I(x - U_{i-1}(x)^T, t - 1) \) in Equation 3.30 is obtained by aligning frame \( F_{t-1} \) with respect to frame \( F_t \) according to the motion field obtained from the previous iteration. This is termed "image warping", which will be explained later. The summation in Equation 3.29 is applied to the region feature's mask in frame \( F_t \). The spatial gradient \( \nabla I = (I_x, I_y) \) is approximated by applying the Sobel operators [46] on frame \( F_t \)

\[
I_x(x, y) = \frac{1}{8}\{I(x + 1, y - 1, t) + 2I(x + 1, y, t) + I(x + 1, y + 1, t) \\
- I(x - 1, y - 1, t) - 2I(x - 1, y, t) - I(x - 1, y + 1, t)\} \\
\]

\[
I_y(x, y) = \frac{1}{8}\{I(x - 1, y + 1, t) + 2I(x, y + 1, t) + I(x + 1, y + 1, t) \\
- I(x - 1, y - 1, t) - 2I(x, y - 1, t) - I(x + 1, y - 1, t)\} \\
\] (3.32)

In our implementation, only intensity value \( I \) is used for motion estimation. For color images, \( I = (R + G + B)/3 \), where \( R, G \) and \( B \) are the intensity values of the three color channels. For a given region feature, a 2-parameter translation (which is a special case of affine motion) is estimated first, the motion estimate is then refined using the 6-parameter affine motion model if the region feature contains more than 250 pixels. This requirement for region size is to ensure sufficient constraints for accurate motion estimation. It is determined empirically and is conservative.

**Image warping**

Image (region) warping is the process of aligning two images (regions) according to the geometric transformation or motion field between them. Assume that we have motion field \( U = (u(x, y), v(x, y))^T \), which is estimated from two consecutive frames, \( F_{t-1} \) and \( F_t \), of a sequence. We can align \( F_{t-1} \) with respect to \( F_t \) according to the motion field as

\[ I_{fw}(x, y) = I(x - u(x, y), y - v(x, y), t - 1) \] (3.33)
3.3 Motion estimation with initial guess

where $I_{fw}(x,y)$ is the warped version of frame $F_{t-1}$. In our case, the motion field is calculated from a set of affine motion parameters, therefore Equation 3.33 is in fact an affine transformation. Because $F_{t-1}$ is warped forward to $F_t$, we call this forward warping. Ideally, $I_{fw}(x,y)$ should be equal to $I(x,y,t)$, but in practice, the two values will not be the same due to the imperfection of motion estimation, noise in intensity values and small variation of intensity values over time.

The intensity value $I_{fw}(x,y)$ in the warped image is determined by

$$I(x - u(x,y), y - v(x,y), t - 1)$$

Depending on the velocity $(u(x,y), v(x,y))$, the coordinates $\hat{x} = x - u(x,y)$ and $\hat{y} = y - v(x,y)$ may take noninteger values. However, in digital images, pixel values are defined only at integer coordinates. Thus inferring what the intensity values at noninteger coordinate locations should be, based only on the pixel values at integer coordinate locations, becomes necessary. The technique used to accomplish this is called intensity interpolation.

The simplest scheme for intensity interpolation is based on a nearest neighbor approach. Given a location with coordinates $(\hat{x}, \hat{y})$, its intensity value is set to be equivalent to that of its nearest integer coordinate neighbor. Although this method is simple to implement, it has the drawback of producing undesirable artifacts, such as distortion of straight edges in images of fine resolution. Smoother results can be obtained by using a more sophisticated technique, such as cubic convolution interpolation [47], which fits a surface of the $(\sin x)/x$ type through a much larger number of neighbors (usually 16) in order to obtain a smooth estimate of the intensity value at any desired point. However, from a computational point of view this technique is costly. A reasonable compromise is to use a bilinear interpolation approach that uses the intensity values of the four nearest neighbors. Assume that the four nearest integer coordinate neighbors of $(\hat{x}, \hat{y})$ are $(x,y), (x+1,y), (x,y+1)$ and $(x+1,y+1)$. This implies that $x \leq \hat{x} \leq x+1$ and $y \leq \hat{y} \leq y+1$. Let $c = \hat{x} - x$ and $d = \hat{y} - y$, then the intensity value of $(\hat{x}, \hat{y})$ is determined as
3.3 Motion estimation with initial guess

\[ I(\hat{x}, \hat{y}) = (1 - c)(1 - d)I(x, y) + c(1 - d)I(x + 1, y) \]
\[ + (1 - c)dI(x, y + 1) + cdI(x + 1, y + 1) \]  

(3.34)

In our implementation, we choose bilinear interpolation for its accuracy and simplicity. \( I(x - u(x, y), y - v(x, y), t - 1) \) is calculated using Equation 3.34.

We can also align \( F_t \) with respect to \( F_{t-1} \) as

\[ I_{bw}(x, y) = I(x + u(x, y), y + v(x, y), t) \]  

(3.35)

where \( I_{bw}(x, y) \) is the warped version of frame \( F_t \). In this case, \( F_t \) is warped backward to \( F_{t-1} \), we call this backward warping.

The "affine flow" algorithm begins with image warping. An approximate motion field is generated for a given region feature according to the corresponding initial guess, which is a 2-parameter translation. This flow field is used to warp the region feature in the first frame toward the second frame (forward warping). Affine parameter estimation and image warping are then repeated iteratively until convergence, i.e. the estimated parameters remain unchanged. Through our experiments, we have found that accurate motion parameters can be estimated for region features with less than 100 pixels. Normally, convergence can be reached within 5 iterations for both the 2-parameter translation estimation and the 6-parameter affine motion estimation, because the initial guess is very close to the true motion parameters.

3.3.3 Robust motion estimation

All the motion estimation techniques discussed so far assume Gaussian noise model and employ least-squares technique to minimize error functions. Although much-used in computer vision applications, least-squares minimization is by nature global, and hence vulnerable to outliers.

The outliers problem is well-known in the statistics literature [49], and arises when a given set of data actually comprises two subsets: a large, dominant subset (the main body of valid data) and a relatively small subset of "outliers" (the contaminants). In the case of parametric motion estimation, the outliers are pixels that undergo motions...
which are not consistent with the dominant motion (the motion of the majority of pixels) in a region. This is usually a result of the imperfection of the static segmentation. Removing outliers is important since an analysis based on both the real data and the outliers distorts conclusions about the underlying process. It is therefore of interest to seek a means of effectively rejecting outliers, thereby restoring the propriety of the data. The task of removing the contaminants is complicated when, as is normally the case, the data in the dominant subset have also been perturbed by noise. The problem of outliers is frequently either ignored or treated heuristically in motion estimation. As a result, the performance of parameter estimation is usually degraded by outliers.

More accurate motion estimation may be obtained by using techniques in robust statistics to reject outliers, as suggested by Black and Anandan [48]. Unlike the standard quadratic error measure \( E(x) = x^2 \), which is a direct consequence of an additive Gaussian noise model, they used the following error measure

\[
E(x) = \frac{-1}{1 + (\frac{x}{\sigma})^2}
\]

(3.36)

where \( \sigma \) is a constant scale factor. This error measure behaves like the quadratic measure when the data errors are small and tends to saturate when the errors become large. So the outliers in the data set are effectively rejected since their influence is close to zero. Such an error function is related to the redescending estimators used in robust statistics, and can be minimized using the deterministic annealing algorithm in which the value of the scale factor \( \sigma \) is gradually decreased.

### 3.4 Feature tracking

In order to obtain a set of extended trajectories of the centroids of region features, feature tracking has to be performed over a certain number of frames. Unlike point tracking, in which case point features are extracted from every frame of a sequence and feature correspondence is performed between every pair of successive frames, in region tracking, once the mask of a region feature has been obtained, it can be tracked
by performing motion estimation and region warping iteratively over several frames. Basically, the region tracking process consists of two steps:

1. Given a region mask in frame $F_t$, estimate its motion between frame $F_t$ and $F_{t+1}$.

2. Warp the region mask in $F_t$ towards $F_{t+1}$ according to its motion parameters, and thus obtain the region mask in $F_{t+1}$.

The above two steps are repeated iteratively over the subsequent frames. The motion estimation step begins with frame 2 and 3, since we have already estimated motion parameters between frame 1 and 2. Again, we use the “affine follow” algorithm to estimate a set of affine motion parameters for each region feature. As mentioned earlier, good initial guess can not only reduce the computational complexity but also improve the accuracy of the estimates significantly. In our implementation, the motion parameters obtained from the previous two frames are used as the initial guess for the current estimation. The effectiveness of such an initial guess relies on the fact that high frame rate (30 frames per second) in video sequences results in small velocity change from frame to frame. Note that this assumption, namely small velocity change, is different from the slow motion assumption, in which case the speed of motion is assumed to be slow.

Once the motion parameters of all the region features have been estimated, their masks in the current frame are warped towards the next frame according to the motion parameters. The same region warping procedure used in Section 3.3.2 is employed here. The difference is that, here, regions' masks, instead of their intensity maps, are warped. A region mask is a binary image, in which a pixel takes a value of 1 if it belongs to the region of interest, and takes a value of 0 otherwise. It provides information about the shape and the location of the contour of a region. Given a region mask in the current frame, the purpose of warping is to determine its location in the next frame. In general, the warped image is not a binary image due to bilinear interpolation. Most of the pixels in this image still take binary values (0 and 1), but pixels around region boundary usually take noninteger values between 0
and 1. In order to obtain a complete region mask, we have to assign each of these noninteger valued pixels an integer value, either 0 or 1. This is actually a contour tracking problem. Here we use the following simple yet efficient rule to determine a pixel’s binary value: if a pixel in the warped image has a value greater than 0.5, it is assigned a value of 1, otherwise, it is assigned a value of 0. This rule is based on the observation that pixels with large values (greater than 0.5) are likely to come from the region of interest in the previous frame. The method is very fast. It essentially requires no extra computation in addition to the standard bilinear interpolation. Our simulation results show that it can track region masks over 10 frames without introducing significant distortion. Even distortion does occur, it tends to erode or dilate region masks isotropically, and therefore does not significantly degrade the accuracy of the position of a region’s centroid, which is important for the following feature clustering process. Figure 3.3 shows the extracted region features and their centroids’ trajectories for the synthetic Two-bars sequence\(^2\). It can be seen that all the features are correctly tracked.

\[\text{Figure 3.3: The extracted region features from the synthetic Two-bars sequence, and their centroids' trajectories over 6 frames (shown in double-length to enhance visibility), with squares indicating features' initial positions.}\]

\(^2\text{The stationary background is also extracted as a feature, with a trajectory of zero length.}\)
3.4 Feature tracking

The number of frames used in feature tracking depends on what type of motions are present in the scene. For sequences containing small or slowly changing motions, region features can be accurately tracked over many frames. But when a sequence contains large motions, deformable motions or motions that change dramatically, features can not be tracked accurately over many frames with the above simple method. In these cases, more sophisticated algorithm, such as the "snakes" proposed by Kass, Witkin and Terzopoulos [26], has to be used. In our current implementation, we just intuitively chose 5 to 10 frames to perform feature tracking. In the future, a confidence measure (such as the motion compensation error over a region) should be employed to let the algorithm automatically decide when to stop tracking (and when to initialize a new round of feature extraction, tracking and segmentation).
Chapter 4

Feature clustering and region classification

4.1 Feature clustering

Once the region features have been tracked over a number of frames so that their centroids form a set of trajectories, the next task is to group these features into a set of clusters that correspond to different objects in the scene. Our feature grouping relies largely on trajectory clustering because feature trajectories reveal important characteristics of different motions in the scene. In many cases, scene structures can also be recovered from a set of sparse feature trajectories. A good example of the usefulness of feature trajectories is the Moving Light Display (MLD) [34]. MLD consists of bright spots attached to the joints of an actor dressed in black, and moving in front of a dark background. The collection of spots carry only two-dimensional information and no structural information, since they are not connected. A set of static spots remain meaningless to observers, while their trajectories create a vivid impression of a person walking, running, dancing, etc. The gender of a person, and even the gait of a friend can be recognized based solely on the trajectories of these spots. In general, features arising from the same object undergo coherent motion and have similar trajectories. On the other hand, features coming from different objects move differently, and the difference between their motions can be easily detected.
by observing their trajectories. Therefore, based on the similarity between feature trajectories, we can group region features into different clusters that correspond to different objects.

### 4.1.1 Clustering concepts

The practice of classifying a data set into different groups is termed "clustering", and is fundamental to many scientific disciplines. Given a $N$-point data set $P$, the objective of clustering is to partition $P$ into $Q$ disjoint non-empty subsets $C_1, C_2, \ldots, C_Q$ ($Q \leq N$), on the basis of some clustering criteria.

There exists a wide range of clustering techniques, differing greatly in their mathematical foundations. A taxonomy of clustering algorithms [35] reveals two primary classes: *partitional methods*, which generate a single partition of the data, and *hierarchical methods*, which build a nested sequence of partitions. Partitional clustering is straightforward in theory: one selects a clustering criterion, evaluates it for all possible partitions, and picks the partition that optimizes the criterion. In practice, it is difficult to select a criterion that successfully translates one's intuitive notion of a "cluster" into a reasonable mathematical formula. Moreover, the number of potential partition is often astronomical even for moderately-sized data sets, making it infeasible to evaluate the criterion over all partitions. These drawbacks make partitional schemes unattractive.

Hierarchical clustering, which is used in our algorithm, arranges the data into a series of partitions, using either *agglomerative* or *divisive* methods. An agglomerative algorithm starts with each point in the data set as an individual cluster (the disjoint cluster), and merges two or more of these singleton clusters to give a second partition based on some affinity measures. This merging process is then repeated, with the number of clusters decreasing until finally a single cluster remains, containing all the nodes (the conjoint cluster). A divisive algorithm performs the procedure in reverse, starting with the conjoint cluster, ends up with the disjoint cluster [36].

The choice of an appropriate affinity measure is crucial to the success of a hierarchical clustering algorithm. Affinity measures are necessarily application-dependent,
e.g. shape and color will obviously be important attributes when sorting apples, bananas and oranges. The properties most appropriate for our domain are those encoding structure and motion of feature trajectories. In many trajectory-based clustering schemes \cite{37}, a feature trajectory is parameterized as a chain vector. Each node in the chain contains a velocity vector (speed and direction) describing the motion of the feature between a pair of successive frames. Parallel trajectories with the same length are grouped together by comparing the corresponding velocity vectors in different chains. However, velocity vectors in a chain do not encode the structure and motion of a trajectory as a whole, this greatly restricts the applicability of the above algorithms.

In this work, we employ a simplified version of the singular value decomposition based trajectory clustering algorithm proposed by Shapiro \cite{22}, which extracts groups of points that maintain their structural integrity over $m$ views (frames).

4.1.2 Trajectory clustering using singular value decomposition

In \cite{22}, Shapiro used a curvature based corner detector to extract point features and track them over several frames to obtain a set of trajectories. He then grouped these trajectories into putative objects using a graduated trajectory clustering algorithm based on singular value decomposition. Following the “least exertion philosophy”, namely “don’t use a more complicated model than necessary”, the algorithm employs a graduated motion analysis scheme, which starts with a simple motion interpretation and gradually increases the complexity of the motion model as needed.

Graduated motion analysis

The following motion models are used in the graduated motion analysis:


   Each trajectory is formed by a 2-D translation of the corresponding feature

   $$ x_i(k) = P_i + d(k) $$

   (4.1)
4.1 Feature clustering

where $x_i(k)$ is the image position of feature $i$ at time $k$, $P_i$ is the reference position for feature $i$, and $d(k)$ is the displacement in frame $k$. They are all 2×1 vectors.

2. Rigid plane undergoing 2-D affine transformation.

Each trajectory is formed by a 2-D affine motion of the corresponding feature

$$x_i(k) = B(k)P_i + d(k)$$

(4.2)

where $B(k)$ is a 2×2 matrix and $\{B(k), d(k)\}$ describes the 2-D affine transformation between the position of feature $i$ at time $k$ and its reference position $P_i$.

3. Rigid object undergoing 3-D affine transformation.

Each trajectory is formed by a 3-D affine motion (projected onto the image plane) of the corresponding feature

$$x_i(k) = M(k)X_i + d(k)$$

(4.3)

where $M(k)$ is a 2×3 matrix, and $X_i$ is the 3-D reference position (a 3×1 vector) of the feature.

Three centroids are defined as follows, which will be used in deriving the affinity measure.

1. Space-centroid of a $N$-point cluster in a single frame

$$\bar{x}(k) = \sum_{i=0}^{N-1} x_i(k)/N$$

(4.4)

2. Time-centroid of a single trajectory over $m$ frames

$$\bar{x}_i = \sum_{k=1}^{m} x_i(k)/m$$

(4.5)

3. Space-time centroid of all $mN$ points

$$\bar{x} = \sum_{i=0}^{N-1} \sum_{k=1}^{m} x_i(k)/mN$$

(4.6)
Affinity measure

The affinity measure encodes the attraction between two sets of trajectories, comprising \( N_a \) and \( N_b \) trajectories respectively \((N_a, N_b \geq 1)\), where \( N = N_a + N_b \) and each trajectory spans \( m \) frames. All of the above models can be formulated in terms of a noisy measurement matrix \( V \) of known rank, whose best estimate \( \hat{V} \) is determined by singular value decomposition (SVD). A suitable “goodness of fit” expression to assess the quality of this estimate \( \hat{V} \) is

\[
a = \frac{\mu_1^2 + \cdots + \mu_p^2}{\mu_1^2 + \cdots + \mu_r^2}
\]

where \( r \) is the actual rank of \( V \), \( p \) is its theoretical rank \((p \leq r)\), and \( \{\mu_1, \ldots, \mu_r\} \) are the \( r \) singular values in decreasing order. This function satisfies our requirements for an affinity measure, with \( a \) varying between 0 and 1 according to the quality of SVD approximation.

Model 1: Pure image translation  The reference points \( P_i \) and the displacements \( d(k) \) are determined by minimizing

\[
\epsilon_1(P, d) = \sum_{k=1}^{m} \sum_{i=0}^{N-1} |x_i(k) - P_i - d(k)|^2
\]

After registering the points with respect to the space-centroids \( \bar{x}(k) \) and \( \bar{P} \), \( \epsilon_1 \) becomes

\[
\epsilon_1(P) = \sum_{k=1}^{m} \sum_{i=0}^{N-1} \|[x_i(k) - \bar{x}(k)] - [P_i - \bar{P}]|^2
\]

The solution is given by a singular value decomposition of the \( 2N \times m \) matrix

\[
V = \begin{bmatrix}
x_0(1) - \bar{x}(1) & x_0(2) - \bar{x}(2) & \cdots & x_0(m) - \bar{x}(m) \\
x_1(1) - \bar{x}(1) & x_1(2) - \bar{x}(2) & \cdots & x_1(m) - \bar{x}(m) \\
\vdots & \vdots & \ddots & \vdots \\
x_{N-1}(1) - \bar{x}(1) & x_{N-1}(2) - \bar{x}(2) & \cdots & x_{N-1}(m) - \bar{x}(m)
\end{bmatrix}
\]
into a rank-1 approximation \((p = 1)\), namely the \(2N \times 1\) and \(1 \times m\) matrices

\[
V \approx LS = \begin{bmatrix}
P_0 - \bar{P} \\
P_1 - \bar{P} \\
\vdots \\
P_{N-1} - \bar{P}
\end{bmatrix}
\begin{bmatrix}
s_1 \\
s_2 \\
\vdots \\
s_m
\end{bmatrix}
\] (4.11)

The single column vector in \(L\) has unit length and due to the centering operation, \(V\) and \(L\) have only \(2(N-1)\) independent rows. If the trajectories in the cluster are indeed formed by a single translational motion, \(V\) has a theoretical rank of 1. However, the rank of \(V\) may be greater than 1 if the single translational motion can not model the motions of all the features in the cluster (inaccurate measurement of feature positions is another factor that can make the rank of \(V\) greater than 1). Therefore, by checking the singular value, we can determine whether the single translation assumption is valid or not. The rank-1 approximation requires \(N \geq 2\) (since \(2(N - 1) \geq 2\)) and \(m \geq 2\) (if \(m = 1\) or \(N = 1\), a perfect solution always exists). The scalar values \(\{s_1, s_2, \ldots, s_m\}\) should all be equal, since all \(P_i\)'s have the same coefficient in \(\epsilon_1\). This equality is enforced by a variance measure. A distance function, measured between the space-time centroids of the two trajectory sets (\(\bar{x}_a\) and \(\bar{x}_b\)), discourages the merging of two distant clusters. The final rank-1 affinity measure is thus

\[
a_1(c_a, c_b) = \frac{\mu_1^2}{\mu_1^2 + \cdots + \mu_r^2} e^{-||\bar{x}_a-\bar{x}_b||^2/\sigma_d^2} e^{-\sum_{k=1}^{m}(s_k-s)^2/\sigma_s^2}
\] (4.12)

where \(\sigma_d\) and \(\sigma_s\) are Gaussian widths, and \(r = \min\{2N, m\}\). This affinity function is bounded \((a_{1,min} = 0\) and \(a_{1,max} = 1\)), and a poor value in any property (large centroid separation, large disparity in \(s\) values, or poor shape correlation) influences the affinity negatively.

**Model 2: 2-D affine transformation** The reference points \(P_i\) and 2-D affine transformations \(\{B(k), d(k)\}\) are determined by minimizing

\[
\epsilon_2(P, B, d) = \sum_{k=1}^{m} \sum_{i=0}^{N-1} |x_i(k) - B(k)P_i - d(k)|^2
\] (4.13)
or equivalently (after registration with respect to the space-centroids)

\[ e_2(P, B) = \sum_{k=1}^{m} \sum_{i=0}^{N-1} |x_i(k) - \bar{x}(k)| - B(k)[P_i - \bar{P}]|^2 \]  

(4.14)

The solution is given by the rank-2 approximation to the \(2m \times N\) matrix

\[ V = \begin{bmatrix} x_0(1) - \bar{x}(1) & x_1(1) - \bar{x}(1) & \cdots & x_{N-1}(1) - \bar{x}(1) \\
 x_0(2) - \bar{x}(2) & x_1(2) - \bar{x}(2) & \cdots & x_{N-1}(2) - \bar{x}(2) \\
 \vdots & \vdots & \ddots & \vdots \\
 x_0(m) - \bar{x}(m) & x_1(m) - \bar{x}(m) & \cdots & x_{N-1}(m) - \bar{x}(m) \end{bmatrix} \]  

(4.15)

whose singular value decomposition into \(2m \times 2\) and \(2 \times N\) matrices is

\[ V \approx LS = \begin{bmatrix} B(1) \\
 B(2) \\
 \vdots \\
 B(m) \end{bmatrix} \begin{bmatrix} P_0 - \bar{P} & P_1 - \bar{P} & \cdots & P_{N-1} - \bar{P} \end{bmatrix} \]  

(4.16)

Note that \(V\) is structured slightly differently from Equation 4.10. The columns of \(L\) are mutually orthogonal unit vectors, the rows of \(S\) are mutually orthogonal, and \(V\) and \(S\) have only \(N-1\) independent columns (due to the centering operation). The theoretical rank of \(V\) is 2. The rank-2 approximation requires \(m \geq 2\) (since \(2m \geq 3\)) and \(N \geq 4\) (since \(N-1 \geq 3\)). The final affinity measure is similar to that in Equation 4.12, except that the approximation is now rank-2 and \(S\) is unrestricted

\[ a_2(c_a, c_b) = \frac{\mu_1^2 + \mu_2^2 + \cdots + \mu_r^2}{\mu_1^2 + \cdots + \mu_r^2} e^{-|x_a - \bar{x}_b|^2/\sigma_a^2} \]  

(4.17)

**Model 3: 3-D affine transformation**  The reference 3-D position \(X_i\) and the 3-D affine transformations \(\{M(k), d(k)\}\) are determined by minimizing

\[ e_3(X, M, d) = \sum_{k=1}^{m} \sum_{i=0}^{N-1} |x_i(k) - M(k)X_i - d(k)|^2 \]  

(4.18)

which becomes (after registration with respect to the space-centroids)

\[ e_3(X, M) = \sum_{k=1}^{m} \sum_{i=0}^{N-1} |[x_i(k) - \bar{x}(k)] - M(k)[X_i - \bar{X}]|^2 \]  

(4.19)
The solution is given by the rank-3 approximation to the $2m \times N$ matrix
\[
V = \begin{bmatrix}
x_0(1) - \bar{x}(1) & x_1(1) - \bar{x}(1) & \cdots & x_{N-1}(1) - \bar{x}(1) \\
x_0(2) - \bar{x}(2) & x_1(2) - \bar{x}(2) & \cdots & x_{N-1}(2) - \bar{x}(2) \\
\vdots & \vdots & \ddots & \vdots \\
x_0(m) - \bar{x}(m) & x_1(m) - \bar{x}(m) & \cdots & x_{N-1}(m) - \bar{x}(m)
\end{bmatrix}
\] (4.20)
computed by singular value decomposition into $2m \times 3$ and $3 \times N$ matrices
\[
V \approx LS = \begin{bmatrix}
M(1) \\
M(2) \\
\vdots \\
M(m)
\end{bmatrix}
\begin{bmatrix}
X_0 - \bar{X} \\
X_1 - \bar{X} \\
\vdots \\
X_{N-1} - \bar{X}
\end{bmatrix}
\] (4.21)

As before, the columns of $L$ are mutually orthogonal unit vectors, the rows of $S$ are mutually orthogonal, and $V$ and $S$ have only $N - 1$ independent columns (due to the centering operation). The theoretical rank of $V$ is 3. Computing the rank-3 approximation requires $m \geq 2$ (since $2m \geq 4$) and $N \geq 5$ (since $N - 1 \geq 4$). The final affinity measure is
\[
a_3(c_a, c_b) = \frac{\mu_1^2 + \mu_2^2 + \mu_3^2 e^{-|z_a - z_b|^2 / \sigma_d^2}}{\mu_1^2 + \cdots + \mu_r^2}
\] (4.22)

The three affinity measures in the graduated clustering algorithm are calculated in sequence, i.e. the algorithm first finds clusters satisfying Model 1, then Model 2, and finally Model 3. In each stage, two clusters are merged if their affinity is larger than a threshold. Each stage completes its growth fully before the next stage begins, with the final clusters of one stage (including singleton points which haven't yet been grouped) serving as the starting clusters for the following stage. This ensures that clusters are always disjoint. Each stage can therefore be considered as a "pre-grouping step" for the next stage.

Modification of the algorithm

The above three-stage clustering algorithm works by fitting a motion model to a set of trajectories to check whether their "shape" can be described by that model
over a number of frames. It performs much better than other trajectory clustering algorithms. In our implementation, only translation model is used for clustering. This is based on our observation that although simple translation can not model complex motions in the scene, in most cases, however, it is sufficient for the reliable clustering of region features’ trajectories. In equation 4.12, the distance function $e^{-|\bar{x}_a - \bar{x}_b|^2/\sigma^2}$ discourages the acquisition of distant features. This property is undesired in our case since two distant regions may belong to one object due to occlusion. Therefore, the affinity measure $a_1$ is modified as

$$a_1(c_a, c_b) = \frac{\mu_1^2}{\mu_1^2 + \cdots + \mu_r^2} e^{-\sum_{k=1}^{m} (s_k - \bar{s})^2/\sigma^2}$$

where the distance function $e^{-|\bar{x}_a - \bar{x}_b|^2/\sigma^2}$ in equation 4.12 has been removed.

The modified clustering algorithm starts with two trajectories, if their affinity is larger than a threshold $T_a$ (say 90%), they are merged as one cluster, otherwise they become two different clusters (two singletons). A new trajectory is tested with every existing cluster, if it can not be merged with any of the existing clusters, it forms a new cluster. This procedure continues until all the trajectories have been processed.

The algorithm works well in most cases. Its main limitation is that cluster growth is monotonic; there is no facility to “split” a group once formed. While overzealous acquisition can be prevented by choosing conservative threshold for the affinity measure, this can also result in many small clusters that refuse to merge. One possible approach to solve this problem is to employ a cluster splitting and merging procedure based on region features’ affine motion parameters, which allows a redistribution of features between clusters, and thus returns wrongly acquired features to their correct clusters.

### 4.1.3 Motion model extraction

After grouping features into different clusters, we derive a set of representative motion models for these clusters by estimating affine motion parameters for each cluster. Motion estimation is applied to every pair of successive frames. This is similar to the affine motion estimation and region feature tracking processes explained in Chapter
3, the only difference is that the parametric motion estimation algorithm is applied to all the regions in a cluster (i.e. the summation in Equation 3.29 is carried out over all the regions in a cluster). For the first two frames, the affine parameters of any feature in a cluster can be used as the initial guess of the motion parameters for that cluster. For the subsequent frames, the previous estimates are used as the initial guess for the current estimation. The motion model extraction process generates a set of likely affine motion models that are exhibited by different objects in the scene. Note that each model contains affine parameters for every pair of successive frames, not just the first two frames.

4.2 Region classification

The final step of our motion based segmentation algorithm is to group regions obtained from static segmentation into different objects using region classification according to the extracted motion models. In region classification, each region is assigned to a motion model that best describes its motion.

4.2.1 Motion model assignment

The basic idea is that if a motion model is a good approximation for a region's motion between two frames, the residual error of that region after motion compensation according the model will be small. We use the Sum of Squared Difference (SSD) of the intensity values between two frames as the measure of residual error. Given a region in frame $F_t$, we align frame $F_{t+1}$ with respect to $F_t$ using backward warping (see Section 3.3.2) according to different motion models. As before, bilinear interpolation is used. For model $i$, we calculate the corresponding $SSD_i$ of a given region as follows

$$SSD_i = \sum_{(x,y) \in r} [I(x,y,t) - I_{bw}^i(x,y)]^2$$

(4.24)

where $(x,y)$ is the position of a particular pixel, $r$ is the set of all the pixels belonging to the given region in frame $F_t$, and $I_{bw}^i(x,y)$ is the resulting intensity value at location $(x,y)$ after warping frame $F_{t+1}$ backward to $F_t$ using model $i$. 
Equation 4.24 is for grayscale images. For color images, the residual error is simply the sum of SSDs of the R, G and B color channels

\[
SSD_i = \sum_{(x,y) \in r} [R(x,y,t) - R_{bu}^i(x,y)]^2 + \sum_{(x,y) \in r} [G(x,y,t) - G_{bu}^i(x,y)]^2 + \sum_{(x,y) \in r} [B(x,y,t) - B_{bu}^i(x,y)]^2 \tag{4.25}
\]

Once the SSDs for all the motion models have been calculated for the given region, we assign a model number to the region as follows

\[
i = \arg \min_i \{SSD_i\}, i = 1, 2, \ldots, N \tag{4.26}
\]

where \(i\) is the assigned model number, corresponding to the motion model that results in the minimum SSD, \(N\) is the total number of models (or clusters). Note that the model assignment is applied to regions in frame \(F_t\).

### 4.2.2 Bidirectional region warping

In the above model assignment process, frame \(F_{t+1}\) is warped backward to frame \(F_t\). For partially occluded regions and regions at frame boundaries, even a correct motion model may result in a large residual error if only backward warping is used. Therefore, false model assignment is likely to occur for these regions. This problem is illustrated in Figure 4.1, in which there are two regions in each of the three consecutive frames: \(F_{t-1}, F_t\) and \(F_{t+1}\). The smaller region is moving to the right and is partially occluded by the larger region, which is moving to the left. If we warp frame \(F_{t+1}\) towards \(F_t\) according to the motion of the smaller region, we will get a large motion compensation error for the smaller region, because a portion of it in \(F_t\) is occluded in \(F_{t+1}\). However, this problem can be solved by align frame \(F_{t-1}\) with respect to \(F_t\) using forward warping. Since no pixel of the smaller region in \(F_t\) is occluded in \(F_{t-1}\), we will get a small residual error when the correct motion model is used for warping. Therefore, for each motion model, both forward warping and backward warping are
4.2 Region classification

Figure 4.1: Three consecutive frames of a synthetic sequence, containing two regions moving horizontally.

performed to calculate two residual errors: $SSD_{fw}$ and $SSD_{bw}$. The smaller of the two is used for model assignment.

In our implementation, forward warping is applied to frame 0 and 1 while backward warping is applied to frame 1 and 2. In both cases, frame 1 is used as the reference frame. After model assignment, each region in frame 1 is labeled with a number corresponding to the motion model that best describes its motion. Regions with the same model number are grouped as one object. The final segmentation result is a grayscale image, in which object masks are depicted with different gray scales.

Figure 4.2: Left: feature trajectories of the synthetic Two-bars sequence over 6 frames (shown in double-length); Middle: object masks (depicted with different gray scales) obtained from region classification using only backward warping; Right: object masks obtained from region classification using bidirectional region warping.
4.2 Region classification

Figure 4.2 shows the region classification results for the synthetic Two-bars sequence. The extracted region features and their trajectories are shown in the left image. These feature trajectories can be easily grouped into three clusters, corresponding to the two bars and the stationary background respectively. The region classification result using only backward warping is shown in the middle image. Two regions along motion boundaries are not correctly classified. The right image shows the classification result using bidirectional region warping. All the regions are correctly grouped according to the three motion models.

Our region grouping process is different from other techniques in the literature. In [15, 16], motion parameters were estimated for all the regions obtained from static segmentation, and clustering based on these motion parameters (e.g. K-means clustering in parameter space) was performed to group regions into objects. The accuracy of motion estimation is crucial to the performance of clustering. However, as mentioned in Section 3.2.4, static segmentation usually results in a large number of small regions. Motion estimation for all these regions is unreliable. Moreover, the number of clusters is not known a priori, making the clustering process extremely difficult. In contrast, we first obtain a few motion models from the extracted region features. Region grouping is then achieved by classifying each region to one of the models based on motion compensation error. The advantage of our method is obvious. Both the number of clusters and the characteristic of each cluster are known a priori. The number of clusters is simply the number of motion models, and each cluster is characterized by the corresponding motion parameters. The task is to assign each region to the motion model that best describes its motion. Such a region grouping process is simple, efficient and robust.
Chapter 5

Results and conclusions

We have implemented the proposed object segmentation algorithm in Matlab on a Sun Sparc Ultra 30 workstation. To illustrate the performance of the technique, we consider a variety of image sequences, containing different types of motions, with both synthetic and real world images and both indoor and outdoor scenes.

5.1 Simulation results

5.1.1 The synthetic Two-bars sequence

The first example is the synthetic Two-bars sequence shown in Figure 3.2. The images are simple, but they are not sufficiently textured, making motion estimation very difficult due to the aperture problem.

The feature extraction, motion estimation, feature tracking and region classification results using our proposed method are shown in Figure 5.1. The top left image shows the static image segmentation result on the first frame, with region boundaries superimposed on the image; The top right image shows the extracted region features; The middle left image shows the feature trajectories over 6 frames (shown in double-length to enhance visibility, with squares indicating features’ initial positions); The middle right image shows the tracked features in frame 6; The bottom left image shows the final segmentation result on frame 1, in which object masks are depicted...
5.1 Simulation results

Figure 5.1: Simulation results for the synthetic Two-bars sequence (see text for detail). First row: static image segmentation result on the first frame and the extracted region features; Second row: feature trajectories over 6 frames (shown in double-length) and the tracked features in frame 6; Third row: final segmentation result.
5.1 Simulation results

with different gray scales; The bottom right image shows the tracked object masks in frame 6. It can be seen that the extracted region features are correctly tracked and the two bars are accurately segmented out. Note that partially occluded regions along motion boundaries are not extracted as features because their sizes change significantly between two frames due to occlusion. This makes the motion estimation and trajectory clustering very reliable.

The true motions in this example are pure horizontal translations of (3, 0) pixels and (-3, 0) pixels per frame for the two bars respectively, and (0, 0) for the background. Our estimated motion models are (3.01, 0.00), (-2.98, 0.02) and (0.01, 0), which are very accurate. In the simulation, we also synthesized the sequence with other horizontal translation parameters for the two bars, ranging from 1 pixels per frame to 6 pixels per frame. Our algorithm generated correct segmentation results as well as motion parameters in all cases. The aperture problem is overcome by our algorithm because good initial guess is available for region based affine motion estimation. Other existing techniques, when applied to this sequence, can not produce the correct segmentation because they fail to estimate motion information accurately.

5.1.2 The Table Tennis sequence

The second example is the grayscale MPEG Table Tennis sequence with image resolution of 352 x 240. Figure 5.2 shows four original frames of this sequence. The ball is very small and is moving with large and changing velocity, ranging from about 13 pixels per frame to about 0 pixel per frame. The racket, the hand and the arm are also moving with different motions against a nearly stationary background.

Figure 5.3 shows the static segmentation results on the first two frames. In each frame, there are more than 180 regions resulted from the Edge Flow segmentation, and they are merged into about 50 regions using intensity-based region grouping\(^1\).

\(^1\)For grayscale sequences, region grouping is based on intensity instead of hue.
Figure 5.2: Four original frames of the MPEG Table Tennis sequence. Top row: frame 1 and frame 2; Bottom row: frame 5 and frame 8.
Figure 5.3: Static segmentation results on the first two frames of the Table Tennis sequence, with region boundaries superimposed on the original frames. Top row: Edge Flow segmentation (left image) and intensity-based region grouping (right image) on frame 1; Bottom row: Edge Flow segmentation (left image) and intensity-based region grouping (right image) on frame 2.
Figure 5.4 shows the results of feature extraction and tracking. Four features, coming from the ball, the racket, the wrist and the table respectively, are extracted. Different motions in the scene are clearly depicted by the feature trajectories. Since there is no ground truth for motion parameters in real world image sequences, a direct evaluation of the estimated motion parameters is impossible. Therefore, we warp the feature masks in frame 1 towards the subsequent frames according to the estimated motion parameters and check how well the warped masks match their corresponding regions. If the estimated motion parameters are accurate, the warped feature masks should match their corresponding regions perfectly. The bottom row in Figure 5.4 shows the warped feature masks in frame 5 and 8. It can be seen that all the feature masks match their corresponding regions very accurately.

![Figure 5.4: Feature extraction and tracking results for the Table Tennis sequence. Top row: the extracted features in frame 1 and frame 2, with their trajectories over 8 frames superimposed on frame 1 (shown in double-length, with squares indicating features’ initial positions); Bottom row: the tracked features in frame 5 and frame 8.](image-url)
5.1 Simulation results

Figure 5.5: Final segmentation result for the Table Tennis sequence. Left: object masks in frame 2, depicted with different gray scales; Right: object boundaries superimposed on the original frame.

Figure 5.5 shows the final segmentation result on frame 2 of the Table Tennis sequence. The scene is segmented into 4 objects, corresponding to the ball, the racket, the hand/arm and the background, with accurate object boundaries. The motion of the left hand at the bottom right corner is different from that of the other hand, but no feature is extracted from the left hand, so it is wrongly grouped with the right hand.

Note that the motion of the ball changes dramatically from frame 1 to frame 8. The traditional two-frame methods will generate different segmentation results if different pairs of successive frames are used. For example, when the ball is at the top of its trajectory (between frame 4 and 5), its motion is nearly zero, and it is hard to segment out the ball from the background. However, when the ball is at the bottom of its trajectory (between frame 1 and 2 or between frame 7 and 8), its motion is quite large and it is easy to segment out the ball using motion information. Therefore, the two-frame formulations might not provide a coherent interpretation of the scene over time. Our method, on the other hand, distinguishes different moving objects according to their motion trajectories over multiple frames. Such a multi-

\footnote{Frame 0 of this sequence is not available, so frame 1, frame 2 and frame 3 are used in the region classification process (see Section 4.2.2), with frame 2 (instead of frame 1) as the reference frame.}
frame formulation can generate more coherent interpretation of the scene.

Figure 5.6: Object tracking result for the Table Tennis sequence, with object boundaries superimposed on the original frames. Top row: the tracked object boundaries in frame 3 and frame 4; Bottom row: the tracked object boundaries in frame 5 and frame 8.

Figure 5.6 shows the result of object tracking according to the extracted motion models. The object masks are warped according to their corresponding motion models from frame to frame using the simple method explained in Section 3.4. Note that each motion model contains a set of affine motion parameters that describe the motion of the corresponding object between every pair of successive frames. It can be seen that most of the tracked object boundaries are very accurate. The boundary of the upper-arm is not accurately tracked because its motion after frame 4 can not be described very well by the motion parameters of the hand/wrist. The left hand at the bottom right corner in frame 1 and 2 (see Figure 5.2) can not be correctly tracked over the following frames because it is wrongly grouped with the right hand.
5.1.3 The Calendar sequence

The third example is the color MPEG Calendar sequence with image resolution of 720×576. The simulation results for this sequence are shown in Figure 5.7, 5.8, 5.9, 5.10 and 5.11.

Figure 5.7 shows the original frame 1 and frame 10 of the sequence. It contains complicated scene with four differently moving objects. A toy train is pushing a ball rolling along the track to the left. The calendar is slowly moving upward with respect to the wallpaper, which appears to be moving to the right because of the camera’s motion (the camera is panning). The calendar’s motion with respect to the wallpaper is almost unnoticeable from two consecutive frames. But it is easy to detect this motion from frame 1 and 10 by examining the relative position of the calendar and the wallpaper.

Figure 5.8 shows the static segmentation result on frame 1. About 400 hundred small regions are generated by Edge Flow segmentation. This number is reduced to about 250 after hue-based region merging. We obtained similar static segmentation result on frame 2, which is not shown here.

Figure 5.9 shows the results of feature extraction and tracking. The difference between the motions of the calendar and the wallpaper is clearly seen from the trajectories of the corresponding features. The tracked feature masks in frame 10 are very accurate as they match their corresponding regions precisely.

Figure 5.10 shows the final segmentation result on frame 1 of the Calendar sequence. All the four moving objects are correctly segmented out and most of the object boundaries are very accurate. As mentioned earlier, the motion of the calendar with respect to the wallpaper is very small, and therefore it is difficult to separate the calendar from the wallpaper according to the motion information if only two successive frames are used. However, when more frames (10 in this example) are used, the difference between the motions of the calendar and the wallpaper becomes obvious and the two objects can be easily segmented out. The shadow of the train on the calendar is grouped with the train because they undergo similar motions. The region
Figure 5.7: Two original frames of the MPEG Calendar sequence. Top: frame 1; Bottom: frame 10.
Figure 5.8: Static segmentation result on frame 1 of the Calendar sequence, with region boundaries superimposed on the original frame. Top: Edge Flow segmentation; Bottom: hue-based region grouping.
Figure 5.9: Feature extraction and tracking results for the Calendar sequence. Top: the extracted features and their trajectories over 10 frames (shown in 4 times of the actual length, with squares indicating features’ initial positions); Bottom: region features in frame 10, after being tracked over 9 frames.
Figure 5.10: Final segmentation result for the Calendar sequence. Top: object masks in frame 1, depicted with different gray scales; Bottom: object boundaries superimposed on the original frame.
Figure 5.11: Object tracking result for the Calendar sequence. Top: object masks in frame 10, after being tracked over 9 frames; Bottom: object boundaries superimposed on frame 10.
5.1 Simulation results

between the ball and the train is also grouped with the train, because this region is not sufficiently textured (due to the shadow) and thus it appears to undergo the same motion as that of the train. The boundaries of the upper part of the calendar are not very accurate because of the imperfection of the static segmentation. A bit of the branch drawn on the wallpaper is attached to the ball for the same reason.

Figure 5.11 shows the object tracking result using the estimated motion models. It can be seen that the tracked object masks are very accurate. There are some false boundaries on the calendar in the bottom image. They are the boundaries of the regions in frame 10 (the dark regions on the calendar in the top image) that are occluded by the train in frame 1. Our simple tracking method can not assign these regions to their corresponding objects. However, the result obtained from our method can be used as the initial condition for more sophisticated tracking algorithms.

5.1.4 The Flower Garden sequence

The fourth example is the grayscale MPEG Flower Garden sequence with image resolution of 352 × 240. Two original frames are shown in Figure 5.12. In this sequence, the camera is translating, introducing strong 3-D parallax in the scene. The tree, the flower bed and the houses move towards the left but at different velocities. Objects closer to the camera (such as the tree) move faster than farther objects (such as the houses).

Figure 5.13 shows the static segmentation result on frame 2. There are about 110 regions generated by the Edge Flow segmentation. Intensity-based region grouping is not very effective for this sequence.

Figure 5.14 shows the results of feature extraction and tracking. The feature trajectories clearly reveal the 3-D parallax of the scene. Note that features from the flower bed are very small in size, but they can be reliably tracked by our algorithm.

Figure 5.15 shows the final segmentation result on frame 2 of the Flower Garden sequence. The image is segmented into 5 objects, lying at different depth in the

---

3False boundaries also exist in Figure 5.6 of the previous example, but they are not shown in order to display the true object boundaries clearly.
scene. Regions of the sky are grouped with the houses because the sky is textureless and it appears to undergo the same motion as that of the houses. The flower bed is segmented into 3 regions because there is a great variation in depth within the flower bed and therefore, its motion can not be described by a single affine model. Some branches of the tree are grouped with regions of the flower bed because their distance to the camera is approximately the same and therefore they undergo similar motions.

Figure 5.12: Two original frames of the MPEG Flower Garden sequence. Left: frame 1; Right: frame 5.

Figure 5.13: Static segmentation result on frame 2 of the Flower Garden sequence, with region boundaries superimposed on the original frame. Left: Edge Flow segmentation; Right: intensity-based region grouping.
5.1 Simulation results

Figure 5.14: Feature extraction and tracking results for the Flower Garden sequence. Left: the extracted features in frame 1 and their trajectories over 5 frames (shown in double-length, with squares indicating features' initial positions); Right: the tracked region features in frame 5.

Figure 5.15: Final segmentation result for the Flower Garden sequence. Left: object masks in frame 2, depicted with different gray scales; Right: object boundaries superimposed on the original frame.
5.1 Simulation results

Figure 5.16: Object tracking result for the Flower Garden sequence. Left: the tracked object masks in frame 5; Right: object boundaries superimposed on frame 5.

Figure 5.16 shows the object tracking result. In the left image, the darkest regions along the boundary of the tree are the regions in frame 5 that are occluded by the tree in frame 2. Their boundaries are shown in the right image. Our algorithm does not include a procedure to determine the depth ordering (or occlusion relationships) for different objects. However, with these disoccluded regions and the extracted motion models over time, it is possible to develop a robust method for the analysis of the depth ordering between different objects.

5.1.5 The Foreman sequence

The fifth example is the color Foreman sequence with image resolution of 176 × 144. Two original frames are shown in Figure 5.17. In this sequence, the head of a talking person is moving, the background is also moving due to camera motion.

Figure 5.18 shows the static segmentation result on frame 2. There are about 110 regions generated by the Edge Flow segmentation, and they are merged into about 70 regions after hue-based region grouping.

Figure 5.19 shows the results of feature extraction and tracking. The motion of the talking person is clearly depicted by the trajectories of the helmet and the face.
**Figure 5.17:** Two original frames of the Foreman sequence. Left: frame 1; Right: frame 10.

**Figure 5.18:** Static segmentation result on frame 2 of the Foreman sequence, with region boundaries superimposed on the original frame. Left: Edge Flow segmentation; Right: hue-based region grouping.
5.1 Simulation results

Figure 5.19: Feature extraction and tracking results for the Foreman sequence. Left: the extracted features in frame 1 and their trajectories over 10 frames (shown in double-length, with squares indicating features' initial positions); Right: the tracked region features in frame 10.

Figure 5.20: Final segmentation result for the Foreman sequence. Left: object masks in frame 2, depicted with different gray scales; Right: object boundaries superimposed on the original frame.
Figure 5.21: Object tracking result for the Foreman sequence, with object boundaries superimposed on the original frames. Top row: the tracked object (the person) in frame 2 and frame 4; Bottom row: the tracked object in frame 6 and frame 10.
Figure 5.20 shows the final segmentation result on frame 2 of the Foreman sequence. The person's helmet and face are grouped as one object as expected. His shoulder, however, is segmented into several regions. The building in the background is also segmented into several pieces, because the motions of the different surfaces of the building cannot be described by a single motion model. One reason for the relatively poor segmentation result is that the features extracted from the person's shoulder and the building are located at object boundaries and frame boundaries. This leads to poor motion estimation for these features. Another reason is the use of affine motion model, which cannot characterize the motions in the scene very well. More sophisticated motion models, such as projective and bilinear transformations, are expected to generate better results.

Figure 5.21 shows the object tracking result. Since the person's shoulder and the building are segmented into several regions, they are not tracked in the simulation. Only one object in the scene (the person's helmet and face) is tracked. The object boundary in frame 10 is not very accurate because affine model cannot describe the person's motion very well.

5.2 Performance and complexity

5.2.1 Performance

Currently, there is no standard procedure to evaluate the performance of an object segmentation algorithm in terms of the accuracy of the resulting object boundaries. Usually, segmentation results are evaluated through visual inspection by the users. It is impractical to implement other existing techniques due to the high complexity of video analysis processes, which, in general, involve numerous steps and requires a lot of optimization and parameter setting through a trial-and-error strategy. Therefore, readers are referred to [12, 15, 16, 54, 55, 56, 57, 58, 59, 60] for a comparison between our results and other published results in the literature. In general, our results are better than those of existing automatic methods, which cannot produce precise object
boundaries or fail to identify all the moving objects in the scene. Actually, our results are comparable to those obtained from semiautomatic methods, in which the objects of interest are specified by the user and the rough positions of object boundaries are obtained manually [57, 58].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>4 for grayscale sequences, and 2 for color sequences.</td>
</tr>
<tr>
<td>$T_h$</td>
<td>$10^\circ$ for color sequences, 5 for grayscale sequences except for the Flower Garden sequence, in which case $T_h$ is set to 1.</td>
</tr>
<tr>
<td>$D_m$</td>
<td>15 pixels for all sequences.</td>
</tr>
<tr>
<td>$T_s$</td>
<td>2% for all sequences.</td>
</tr>
<tr>
<td>$T_a$</td>
<td>90% for all sequences except for the Flower Garden sequence, in which case $T_a$ is set to 75%.</td>
</tr>
</tbody>
</table>

*Table 5.1: Parameter setting for the simulations*

In the simulations, several parameters need to be set by the user. These parameters are:

- The image scale $\sigma$, which is used in Edge Flow segmentation.
- The threshold $T_h$ for the (maximum) difference between two regions’ average hue values\(^4\), which is used in hue-based region grouping.
- The expected maximum displacement $D_m$ of an object between two successive frames, which is used in feature correspondence.
- The threshold $T_s$ for the (maximum) difference between the sizes of a pair of corresponding region features, which is used in feature correspondence.
- The threshold $T_a$ for the (minimum) affinity of two sets of trajectories, which is used in trajectory clustering.

\(^4\)For grayscale images, $T_h$ is actually the threshold for the (maximum) difference between two regions’ average intensity values (see Section 2.3 for detail).
5.2 Performance and complexity

Table 5.1 summarizes the parameter setting for the simulations. Currently, there is no good method to determine these parameters automatically, they have to be set by the user using a trial-and-error strategy. However, it can be seen from Table 5.1 that most of the parameters are quite general for all the sequences.

5.2.2 Complexity

Because of the high complexity of the object segmentation process, a thorough analysis of the computational complexity of the proposed algorithm in terms of the number of basic operations (multiplications) is extremely difficult, therefore we only evaluate the computational complexity in terms of computation time. However, it should be pointed out that computation time not only depends on the algorithm itself but also depends on the implementation of the algorithm and the computer used.

For the grayscale Table Tennis sequence and Flower Garden sequence with image resolution of $352 \times 240$, the whole segmentation process takes about 4.5 minutes on a Sun Sparc Ultra 30 workstation. The computation time for the Edge Flow segmentation\(^5\) is about 3 minutes (recall that we need to segment the first two frames of a sequence using the Edge Flow algorithm). The rest of the process (including intensity-based region merging, feature correspondence, motion estimation/feature tracking, trajectory clustering and region classification) is relatively fast. It is implemented in Matlab and takes about 1.5 minutes. The computation time for the color Calendar sequence with image resolution of $720 \times 576$ is about 31 minutes, 28 of which are taken by the Edge Flow segmentation. The rest of the process takes about 3 minutes. The total computation time for the color Foreman sequence with image resolution of $176 \times 144$ is about 5 minutes. As can be seen, our algorithm is computationally expensive, but a large fraction of the computation time is taken by the Edge Flow segmentation, which also requires a huge amount of memory. In our simulation, we have to divide a $720 \times 576$ image into two parts and segment them separately, because the Sun Sparc Ultra 30 machine (with 128MB ram and 96MB swap

\(^5\)We obtained the binary code for the Edge Flow segmentation algorithm from Dr. W.Y. Ma at HP Labs and Professor B.S. Manjunath in the ECE Dept. at UCSB.
space) does not have enough memory to segment the whole image using the Edge Flow algorithm. The overall computational complexity of the proposed algorithm will be reduced significantly if other simple yet robust static segmentation algorithms are available.

Currently, our algorithm is not suitable for real time application. In fact, a major challenge for the object based video coding standard is that a real time encoding process is far from reality. Matsushita Electric Industrial Co. Ltd. recently demonstrated its MPEG-4 video compression algorithms. It takes about an hour to convert a 30-second video footprint into MPEG-4 format, even though different objects in the original footprint have already been prepared as separate component data.

5.3 Discussions and future directions

The proposed object segmentation algorithm has several limitations, which are also important issues that should be addressed in future research:

1. Our method relies on the extraction of region features from every object (including the background) in the scene. A moving object may not be segmented out if no feature can be extracted from that object, as the case of the left hand in the Table Tennis sequence (see Section 5.1.2 and Figure 5.2). One possible way to solve this problem is to modify the region classification procedure: if a region has large motion compensation errors for all the motion models, it should not be assigned to any of the models, instead, a new motion model should be estimated for that region. Another effective approach to solve this problem is to allow certain degree of user supervision for feature extraction. Researchers have pay less attention to semiautomatic methods than to automatic methods because user supervision is generally considered as a negative characteristic of a segmentation algorithm. It is impractical to incorporate user supervision in most real time applications. However, many multimedia applications allow off-line operations. In those situations, proper user supervision can not only improve the accuracy of the final result but also reduce the computation time
significantly, because while many image analysis tasks are extremely difficult even for today's most powerful computers, they can be done by a human user effortlessly. It should be pointed out that a common drawback of many automatic methods (including our proposed method) is the dependence of their performance on a lot of parameters that need to be set by the user, usually through a trial-and-error strategy. This fine-tuning process can be considered, to a great extent, as a form of hidden supervision [57].

2. In our algorithm, motion information is used only for region grouping. The accuracy of the segmented object boundaries mainly depends on the performance of static image segmentation. Methods that use motion information to refine object boundaries should be developed. One possible way to achieve this is to use temporal integration [8], in which object boundaries are refined over time by assigning image pixels to different objects according to motion information during the object tracking process. Another way is to incorporate motion and static information into a statistical framework [19, 20], which is a promising approach to combine multiple cues in an image sequence, such as color, motion and pixel position, to perform object segmentation. Color and motion are employed at different stages in our algorithm, with motion as the main criterion for object segmentation. This leads to a common problem of motion based segmentation algorithms: an object of interest can not be segmented out using motion information if it does not move with respect to the background (or other objects). The table in the Table Tennis sequence (see Figure 5.5) is such an example: although it can be reliably segmented out using static segmentation, it is eventually grouped with the stationary background because its motion is nearly zero. The statistical approach mentioned above might be a solution to this problem.

3. Our algorithm can not cope with objects undergoing complex motions. Human movements, for example, consist of motions of different body parts which can not be described by a single motion model. Therefore, a walking or running person is likely to be segmented into several parts by our algorithm. To solve this problem, high level
motion recognition and object recognition algorithms should be used. Our algorithm can be employed as a preprocessing for these more sophisticated techniques. Finally, this thesis concentrates on the problem of object segmentation, another important issue is object tracking, i.e. tracking, refining and updating a given set of object masks over time. The main difficulty for object tracking is how to cope with complex object motions and occlusion/disocclusion. Moreover, the depth ordering of different objects has to be determined in order to obtain a complete scene description. This is also a topic for future research.

5.4 Conclusions

In this thesis, we addressed the problem of object segmentation in image sequences. We designed and implemented a novel automatic, multi-frame, region-feature based motion segmentation technique, which exploits both static and motion information to precisely localize object boundaries. The first two frames of an image sequence are segmented using static information. Salient region features are extracted from the segmented images and are tracked over a number of frames. Trajectory clustering is then performed to group these features into putative objects, from which a set of motion models are estimated. Final segmentation result is obtained by region classification based on the estimated motion models. The proposed technique combines the advantages of feature based methods and gradient based methods. Experimental results on synthetic and real world image sequences show that our algorithm can correctly identify moving objects in a video sequence and accurately segment them out. Several issues should be considered in the future in order to improve the performance of our algorithm and enable it to cope with a wider variety of situations.
Bibliography


BIBLIOGRAPHY


