December 16, 2005

To the Graduate School:

This thesis entitled "Motion Segmentation at Any Speed" and written by Shrinivas J. Pundlik is presented to the Graduate School of Clemson University. I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

Dr. Stanley Birchfield, Thesis Advisor

We have reviewed this thesis
and recommend its acceptance:

Dr. Ian Walker

Dr. Adam Hoover

Accepted for the Graduate School:

MOTION SEGMENTATION AT ANY SPEED

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

by
Shrinivas J. Pundlik
December 2005

Advisor: Dr. Stanley Birchfield

ABSTRACT

Common fate, or common motion, is a very strong cue for segmentation, and hence motion segmentation is a widely studied problem in computer vision. Most previous segmentation approaches deal with motion either between two frames of the sequence or in the entire spatio-temporal volume. In this thesis an incremental approach to motion segmentation is presented that allows for a variable number of image frames to affect the decision process, thus enabling objects to be detected independently of their velocity in the image. Feature points are detected and tracked throughout an image sequence, and the features are grouped using a region-growing algorithm with an affine motion model. The algorithm uses a single parameter, which is the amount of evidence that must accumulate before features are grouped. Procedures are presented for grouping features, measuring the consistency of the resulting groups, assimilating new features into existing groups, and splitting groups over time. Experimental results on a number of challenging image sequences demonstrate the effectiveness of the technique.

DEDICATION

This thesis is dedicated to my mother and father who have always been supportive in my endeavors.

ACKNOWLEDGEMENTS

I am indebted to my advisor, Dr. Stan Birchfield, for his guidance, encouragement and support throughout the course of this work.

I would like to thank Dr. Ian Walker and Dr. Adam Hoover for agreeing to be part of my thesis committee and for their helpful suggestions.

Finally, I would like to thank all my friends and lab mates who made my stay at Clemson a memorable experience.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF FIGURES

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

CHAPTER 1
INTRODUCTION

An insight provided by the Gestalt psychology in the field of human visual perception has led researchers to develop image and video segmentation systems that delineate homogeneous image regions based on the Gestalt laws of grouping. The Gestalt theory of visual perception argues that human visual perception is more adept at focusing on well-organized patterns rather than on disparate parts, implying that grouping, or clustering, forms the fundamental idea behind visual interpretation. In the human visual system, representation of an object is a result of grouping of individual neural responses, which in turn is guided by factors underlying the scene such as similarity between the elements, closure, symmetry, proximity, or common fate. These criteria for grouping are summarized as the Gestalt laws of grouping.

An image can be effectively segmented using one or a combination of many different criteria such as proximity, similarity, symmetry, or closure. For example, similarity of color or intensity values of individual pixels can be used as one criteria of segmentation of an image. At the same time, use of the proximity criteria rules out grouping together pixels or regions of similar intensity far apart from each other in an image. Segmentation of videos is a little more complicated process since videos are sequences of individual frames and time becomes an important consideration. In such a situation, the criterion of common fate, or common motion, is of special significance because it deals with the change in the scene due to motion over time. Therefore, motion segmentation can be defined as the grouping or clustering of homogeneous image regions based on the criterion of similar motion.

Motion segmentation is important because it forms the basis of algorithms dealing with object detection, tracking, surveillance, robot motion, image and video compression, and shape recovery, among others. At the same time it is also a challenging task for variety of reasons. First, the 3D motion in the scene is mapped to a 2D image plane making the problem of quantifying image motion and subsequent recovery of the motion parameters an under-constrained problem requiring additional constraints to be placed on the motion of

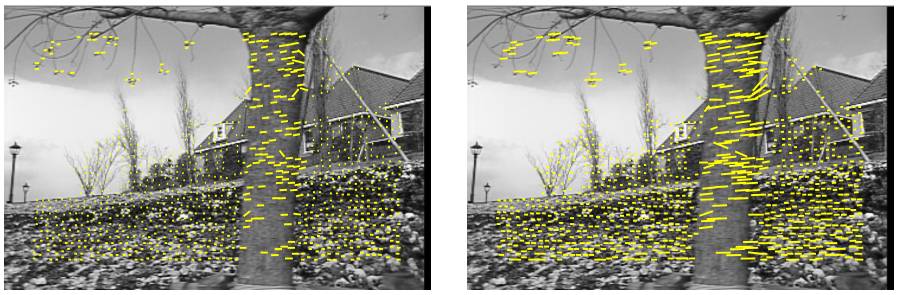

Figure 1.1 Motion fields generated by sparse feature points distributed over the image. LEFT: Motion between the first and the second frames of the sequence. RIGHT: Motion between first and the third frame.

image regions. Image noise adds to the ambiguity to the motion parameters. For example, noise may result in the change in the pixel values in subsequent frames even if there is no motion between the two frames. Occlusions or disocclusions may occur in the sequence when an object moves into the background, gets occluded by the foreground object and then reappears into the foreground. This poses a challenge to accurate association of moving regions through the sequence.

The two basic steps in a generic motion segmentation procedure are (1) determining the motion vectors associated with each pixel assuming that the intensity of that pixel does not change in the subsequent frames, and (2) clustering pixels that exhibit common motion. Instead of finding motion vectors associated with each pixel, we can find key feature points in the image and track those over the sequence to represent motion over an image sequence. The resulting motion field looks similar to that generated using a dense optical flow procedure (see Figure 1.1). Usually, motion field in an image refers to the true motion of a pixel but in the present case it represents the motion of a feature point between two frames. Grouping the features according to their motion is then equivalent to clustering individual pixels and can be done in multiple ways, for example, by setting a global motion threshold so that features with motion above the threshold value fall in one cluster and the rest in another cluster, or by defining a threshold on the error in motion for a cluster of features such that features with motion within the set error range are grouped together.

This thesis proposes an incremental approach toward motion segmentation, where segmentation of image regions is done when sufficient evidence has accumulated over time.

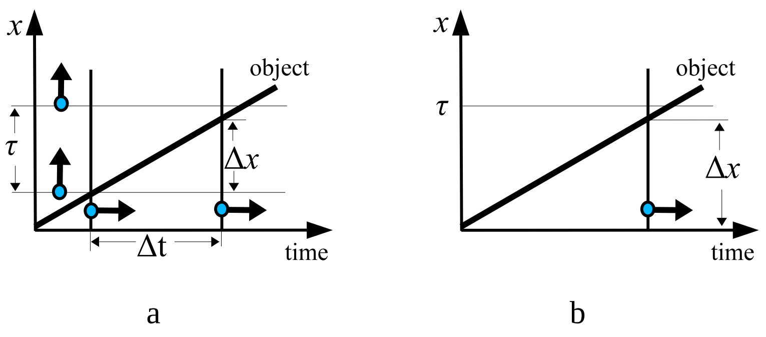

Figure 1.2 An object moves at constant speed against a stationary background. LEFT: If a fixed number of image frames are used, then slowly moving objects are never detected. RIGHT: Using a variable number of images enables objects to be detected independently of their speed.

This is in contrast to most of the existing motion segmentation algorithms that process over either two frames at a time or a spatio-temporal volume of a fixed number of frames assuming the motion of an object to be linearly changing or constant through the sequence. Such algorithms fail to handle non-uniform motion of objects through the sequence. The proposed approach enables motion segmentation over long sequences with multiple objects in motion.

Incremental Approach to Motion Segmentation

One prominent challenge while grouping features is to reliably infer the velocities of feature points of a sequence from the positions of the features in the corresponding image frames due to tracking errors and image noise. A related problem is the accumulation of sufficient evidence through the sequence to enable an accurate segmentation of the scene. The evidence for segmentation in the present context is the motion of features in the sequence. Since the features associated with the objects in the scene may move with different velocities, accumulation of evidence for segmentation may require different number of frames for different features at different instances of time. Accumulating an arbitrarily large number of frames may not guarantee a reliable solution as the feature tracks may vary unpredictably through the sequence.

Traditional approaches toward motion segmentation have been to consider image motion between two frames of a sequence at a time and perform motion segmentation by setting a threshold on the velocities of the regions (see Figure 1.2a). If a region is moving with a velocity less than the threshold then it may never be detected. An alternative is to set a fixed threshold on the displacement of the image regions. The advantage of this approach is that if we wait for a sufficient time then regions moving with a very small velocity can be also detected (see Figure 1.2b). So in such a situation the segmentation is related to the process of accumulating evidence of motion. While considering image motion in two frames, instead of segmenting the whole scene, we wait until we have sufficient evidence to segment each region. As a result, different objects are segmented from the scene over a period of time depending upon their motion in the image with respect to the same reference frame. Once the initial segmentation is achieved over a period of time, the individual image regions are tracked and their motion models are updated while including the new information that appears in the scene. This addresses the problem of successfully segmenting long image sequences with multiple image regions in non-uniform motion.

Previous Work

Different techniques exist to quantify the image motion and perform motion segmentation based on such a measure. There are two approaches for motion segmentation. One relies on the low level image data to find the motion of each pixel and then tries to combine this information to come up with some kind of estimate of image motion as a whole. This is termed as a bottom-up approach in the field of computer vision. The other approach, which is known as a top down approach, tries to label different image regions in motion using some a priori information or some high level model.

Some of the earlier approaches use optical flow fields to describe the 2D velocity of the pixels of the image. A smoothness constraint is imposed on the velocities of the neighboring pixels. This kind of approach is fraught with the problems like dealing with large untextured areas in an image or effective imposition of the smoothness constraint so as to not smooth over the discontinuities.

More recent approaches allow simultaneous estimation of multiple global motion models with a layered representation of the image [15], [1], [16], [17], [19], [18], [9]. Earliest of such approaches was proposed by Wang and Adelson [15]. They estimate the optical flow and group pixels using an affine motion model iteratively such that a model corresponding to each layer in the sequence is obtained. Subsequent papers on the layered motion model by Weiss [16], Weiss and Adelson [17] assume a set of parametric motion fields exist in the motion sequence, each represented by a probabilistic model while the overall motion in the sequence is given by a mixture model. To find motion at an individual pixel, a particular motion field set is determined and a sample is drawn from it. In this case both, the parameter values of the motion field and the motion field associated with each pixel are unknown. This is a missing data problem and the expectation maximization (EM) algorithm is used to iteratively determine the motion field an individual pixel belongs to and the parameters of those motion fields [6]. Layer based motion segmentation using graph cuts is described in Xiao and Shaah [19], Wills et. al. [18] while Ke and Kanade [9] use a subspace based approach, originally proposed in [5], to assign pixels to different layers. Layered motion model approach of motion segmentation are effective in demarcating the motion boundaries and allow reconstructing the motion sequence by manipulating the individual motion layers. Limitation of this approach lies in the initialization phase, where one has to decide beforehand the nature of the motion models. Motion segmentation using multiple cues such as color, spatial relationships and motion is discussed in [10].

A fundamentally different approach based on normalized cuts algorithm was proposed by Shi and Malik [13]. The motion segmentation is posed as a graph partitioning problem in a spatio-temporal volume and solved as a generalized eigenvalue problem. A weighted graph is constructed with each pixel as a node connected to pixels in the spatio-temporal neighborhood and segmented using the normalized cuts algorithm. The normalized cuts framework splits the weighted graph constructed earlier in a balanced manner instead of splitting small isolated regions. The normalized cuts algorithm outputs a matrix containing the weights for association of each node to the different clusters. The approach is computationally expensive.

Image motion leads to such interesting phenomena as motion discontinuities and occlusion boundaries in the image. Different image patches moving with different velocities give rise to motion discontinuities as some image regions may move in front of others, occluding them. The boundary of the foreground region forms the occlusion boundary between it and the occluded region. These motion discontinuities or occlusion boundaries through a sequence can lead to reliable motion segmentation of the scene. Black and Fleet [4] describe an algorithm for detection and tracking of motion discontinuities using the condensation algorithm which is extended in [11] by incorporating edge information along with the motion information for better motion boundary localization.

As an alternative to dense optical flow, motion between two frames of a image sequence can be represented by detecting and tracking point features between the frames. The features are small image regions that exhibit high intensity variation in more than one direction [14]. Some work [2],[8] involves detecting and tracking such point features and grouping them based on motion in case of vehicle tracking scenario. Segmentation and tracking of human limb motion using robust feature tracks is described in [7]. Approach described in [3] finds good features in the sequence and groups them iteratively using an affine motion model.

Overview

The work presented in this thesis is based on the algorithm proposed by Birchfield [3] for detecting motion discontinuities between two frames of a sequence. Point features are detected and tracked through the sequence and motion between two frames is considered to group the point features according to an affine motion model to find different image regions in motion. Using this two frame motion segmentation algorithm, segmentation of the whole scene is done incrementally, over a period of time. The evidence of segmentation is calculated by a feature group consistency criterion that measures the consistency, or similarity, of feature groups formed under varying conditions. Motion models are computed for each of the feature groups and the groups are tracked through the sequence while continuously updating the motion model to accommodate scene changes occurring over time.

To summarize, this thesis describes:

1. various methods for grouping features based on their motion in two frames of a sequence,
2. different parameters that affect the feature grouping process, and
3. an algorithm to extend the two frame motion segmentation algorithm to work with multiple frames.

Outline

The rest of the text is organized as follows. In Chapter 2 detection and tracking of features in a video sequence is described. Two different methods of feature grouping namely, clustering and region growing are explained in Chapter 3 and Chapter 4 respectively. In Chapter 5, the feature group consistency criterion and maintenance of the feature group is described. Experimental results on variety of image sequences are demonstrated in Chapter 6. Finally, conclusions and future work are presented in Chapter 7.

CHAPTER 2
FEATURE TRACKING

Any algorithm dealing with motion segmentation has to tackle the problem of quantifying the image motion. An ideal measure of image motion accurately identifies image regions undergoing different motion in varying circumstances. This means, it allows efficient recovery of the motion parameters and helps in establishing correspondence between the subsequent frames in an image sequence. It builds the motion hypothesis based on the low level operators, i.e., it follows a bottom-up approach. This makes it general enough to be applied to different kinds of images in different circumstances. Point features, though not an ideal measure, are sufficient to represent image motion in this thesis. This chapter describes the motivation behind selecting point features for motion representation, the algorithm to detect and track the features through an image sequence and its limitations. It further describes a method to improve the performance of the feature tracking algorithm while dealing with a long sequence.

Motivation

There are two principal ways to represent image motion. One approach is to use dense motion fields, i.e., to find motion vectors associated with each pixel in an image while the other approach is to detect key feature points in the image and find motion vector of each feature point. The main motivation behind using sparse feature points spread over the image instead of a dense motion field representation is to reduce the time and complexity of the computation. Since in most cases the image regions under motion are much larger than a pixel, a dense motion field has a lot of redundant information. The same information can be reliably obtained from a large number of features spread over the image with less computational cost. For example in the experiments described in this thesis, 1000 features are enough to cover an image of size 320×240 instead of 76,800 motion vectors.

Feature points are not individual pixels but a neighborhood of pixels taken together. They capture the intensity variation in this neighborhood and are reliable low-level operators making them repeatable under varying circumstances (for example, between subsequent frames of an image sequence). Also, establishing correspondence between features in two frames becomes easy and for this reason they are well suited for the process of tracking. Since they consider a neighborhood of pixels, the effect of noise is reduced. To summarize, motion of feature points through the sequence gives an effective representation of image motion and hence, detecting and tracking feature points forms the very basis of the proposed motion segmentation algorithm.

Feature Tracking Algorithm

Features are detected and tracked over the image sequence by the Kanade Lucas Tomasi (KLT) feature tracker based on the algorithm described in [14]. The dissimilarity of feature windows in two images is given by

$$\epsilon = \int\int_W [I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2 d\mathbf{x} \tag{2.1}$$

The idea behind tracking features is to minimize the error or the dissimilarity shown above. The algorithm chooses good features to optimize the performance of the tracker. Good feature points are those which correspond to real world points. Consider an image sequence $I(x,t)$ where, $\mathbf{x} = [\mathbf{u}, \mathbf{v}]^{\mathbf{T}}$ are the coordinates of any image pixel. During tracking if it is assumed that intensities of the points in images remain unchanged then

$$I(\mathbf{x}, \mathbf{t}) = \mathbf{I}(\delta(\mathbf{x}), \mathbf{t} + \mathbf{t}') \tag{2.2}$$

where δ refers to the motion field. If the image pixels are assumed to be translating then the motion field is specified by $\delta = \mathbf{x} + \mathbf{d}$ where $\mathbf{d}$ is the linear displacement vector. The aim now is to find $\mathbf{d}$ such that, it minimizes the Sum of Squared Distances. The resultant system is

$$G\mathbf{d} = \mathbf{e}, \tag{2.3}$$

where

$$G = \sum_{(u,v)\in W} \begin{pmatrix} I_u^2 & I_uI_v \\ I_uI_v & I_v^2 \end{pmatrix}, \mathbf{e} = -t' \sum_W I_t [\, I_u \quad I_v \,]^T$$

W is the size of the window, $[\, I_u \quad I_v \,] = \nabla I = [\, \partial I/\partial u \quad \partial I/\partial v \,]$ and $I_t = \partial I/\partial t$.

Finding Good Features

The matrix G in 2.3 is given by

$$G = \int\int_W (\nabla I)(\nabla I)^T w(\mathbf{x}) d\mathbf{x} \tag{2.4}$$

For a given image, the first step is to find the gradient for every coordinate point (i.e., each pixel). But individual pixels are not of any use for tracking over a sequence. Hence, collection of pixels called feature window are tracked. Neighborhood pixels combine to form a feature window. Integrating over this feature window gives us a matrix G, whose eigenvalues are of primary interest. Corresponding eigenvectors give the principal components (direction) while the eigenvalues decide the amount of scaling of these vectors in their respective directions. Since the G matrix consists of spatial gradient information, its eigenvectors and eigenvalues give the principal direction of change in the intensity and the amount of change respectively. In general, a good feature is one which has its smallest eigenvalue greater than the user defined threshold. If λ_1 and λ_2 are two eigenvalues of G for a feature window such that $\lambda_1 > \lambda_2$ and δ_{feat} is the user defined threshold on the eigenvalue, then feature is considered a good feature if $\lambda_2 > \delta_{feat}$.

Significance of having two large eigenvalues for the G matrix of a feature window is that it shows the variation of intensities in more than one direction in the feature window. Such locations in an image are important as they are the easiest to search in the subsequent frames if their motion could be estimated. If both λ_1 and λ_2 are below the threshold δ_{feat}, then the feature window has uniform intensity and does not carry any significant information which could identify the feature. As a result, texture-less regions in an image do not have any good features. This can be seen form Figure 2.1 (top row), where good features are not detected in the clear sky. If one of the eigenvalues is very large as compared to the other then it indicates a strong gradient in one direction, i.e., presence of an edge. Such a feature window is not considered good as the location of the feature becomes ambiguous in the subsequent frames.

As a result of feature tracking step, tracks of the feature points are obtained over the required number of images. The detected features in the initial frame are shown in Figure 2.1 for two different sequences. Figure 2.1 (top-right and bottom-right) shows the features

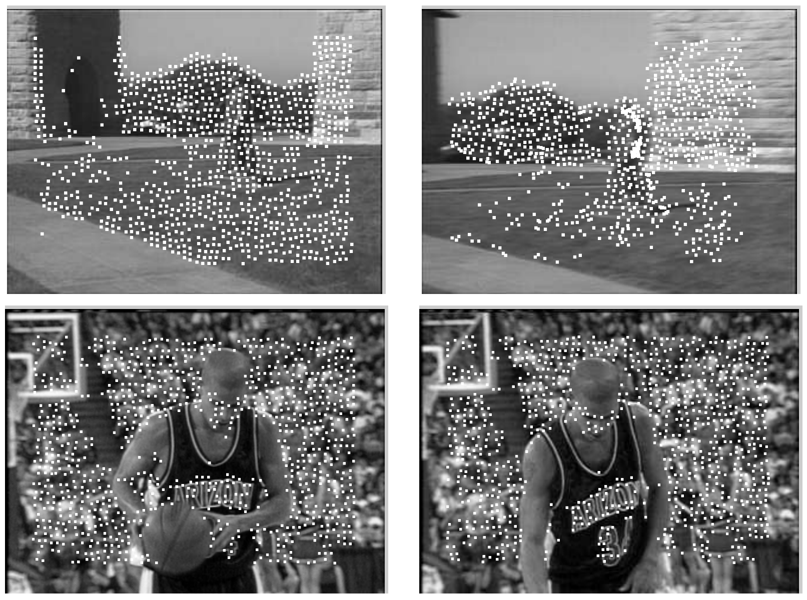

Figure 2.1 Detection and tracking of features for real world sequences. Detected features overlaid on the first image and features tracked through the nth frame of the sequences.TOP: The statue sequence. BOTTOM: The freethrow sequence.

tracked from the initial frame through the final frame. As it can be seen some features are lost while some new features are added as new scene appearers in the sequence.

Limitations of the KLT Feature Tracker

Certain assumptions made while developing the original KLT algorithm limit the performance of the feature tracker. KLT matches images in successive frames. This makes the implementation easier but any feature information in previous frames is ignored. It cannot model complex variations in the image region over a long sequence as this information cannot be obtained only from displacement parameters.

It also assumes that there is no occlusion of the feature window in the next frame. Any occlusion results in the loss of that feature altogether. Even if the feature reappears after a few frames, there is nothing to establish correspondence between the lost and reappeared feature. There also exists a problem about features straddling a depth discontinuity or a boundary making such features unreliable for tracking. But still, depth discontinuities and boundaries have wealth of information regarding the object being tracked, which KLT cannot effectively utilize. Size of the tracking window plays an important role. If the size of feature window is increased, then the motion variation information can be reliably calculated but the risk of the feature drifting along with a boundary becomes more.

KLT considers each feature as an independent entity while tracking but in may situations the features may follow some pattern where they are not independent. There are other issues such as KLT assuming constant intensity levels over the successive frames and a simple (rigid) geometry while calculating the transformations. Its performance deteriorates when some image regions undergo non-rigid transformation.

Affine Consistency Check

In the feature tracking algorithm described above, the displacement $\mathbf{d}$ is calculated using a translational model and dissimilarity between a feature window in successive frames is given by equation 2.1. While dealing with long image sequences, a feature window may undergo significant changes and its appearance may become different as compared to its original form in the frame in which it was detected. This leads to tracking failures such as

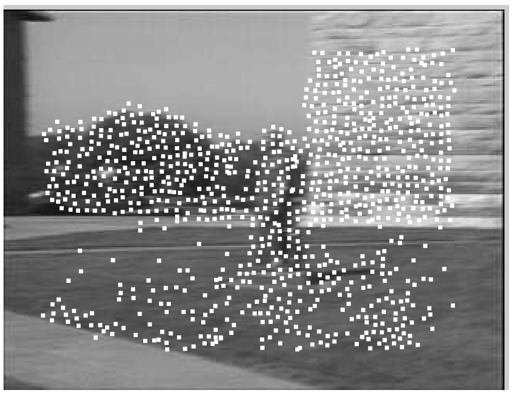
Figure 2.2 Feature points obtained after the affine consistency check for the statue sequence.

features straddling discontinuities or bunching of features near occlusion boundaries. This effect can be seen in Figure 2.1 (top-right) where features are bunched together on the right side of the statue.

Shi and Tomasi [14] describe an approach that measures the dissimilarity between a feature in the current frame and the corresponding feature in the first frame by using an affine motion model. Equation 2.1 can be modified to include the affine parameters as

$$\epsilon = \int\int_W [I(\mathbf{x}) - J(A\mathbf{x} + \mathbf{d})]^2 d\mathbf{x}$$

which yields a system of linear equations pertaining to the six affine parameters solving for which gives the distortion of the feature. Features are tracked in successive frames using a translational model while comparison of features in the current frame and initial frame is done using the affine model. If the dissimilarity exceeds the threshold value then the feature is discarded. Feature tracking using affine consistency check is shown in Figure 2.2 and it can be seen that the features are evenly distributed and there is no bunching of features at the motion discontinuities. This is incorporated in the latest version of KLT.

CHAPTER 3
FEATURING CLUSTERING

Once features are detected and tracked, the next step is to group all the features exhibiting similar motion. This chapter describes some common clustering algorithms such as K-means and normalized cuts for clustering features. Although the clustering techniques are successful in grouping features in some simple image sequences, they have limitations while handling long sequences. This chapter is just an exploration of various ideas and the techniques described in it are subsumed by the next two chapters.

Introduction to Clustering

Clustering can be defined as the process of partitioning or segregation of a data set into smaller subsets or clusters in a manner such that the elements of individual clusters share a predefined similarity criterion. For images, in most cases the data set for clustering consists of individual pixel values or vectors representing the color, coordinates of the pixels, the texture around the pixel, or some other properties calculated from the image data. For a typical clustering method, a data set X with its elements given by x_i, is partitioned into N clusters C_1,C_2, $\ldots$,C_N with means μ_1,μ_2, $\ldots$,μ_N. Then an error measure that represents how well the data is clustered can be defined using least squares:

$$e = \sum_{k=1}^{N} \sum_{x_i \in C_k} \| x_i - \mu_k \|^2 \tag{3.1}$$

In many implementations of clustering techniques, the number of clusters N to be found for a particular data set, is known beforehand. This enables the user to define an objective function that can be optimized to obtain better clustering results. If N is not known, then the optimal number of clusters required can be found out from the partitioning of the data itself for example by setting a threshold on the variance of each cluster. Different techniques for data clustering exist such as, K-means, agglomerative and divisive clustering, histogram mode seeking, graph theoretic and so on.

Feature Clustering Using K-means

K-means is a very commonly used technique for data clustering. Before the clustering process starts, the algorithm requires a prior knowledge of the clustering space and the number of clusters to partition the data in. Usually, the clustering space depends on the dimensionality of the data elements. For images it may be a 1D space if only the pixel intensities are used for classification, 2D if pixel coordinates are used or of multiple dimensional space if each pixel is associated with a high dimensional vector describing local texture.

A simple form of K-means algorithm to cluster points in a 2D plane is discussed below. It involves minimizing an objective function which is similar to the error measure for clustering as shown in 3.1. A direct search for the minimum of such a objective function is not feasible as number of points may be large. Hence to tackle this situation, the algorithm iterates through two steps: assuming the cluster means, perform the allocations of the data points and based on the point allocations in the previous steps, calculate the cluster means till convergence. Convergence is achieved when the error value defined in 3.1 reaches below a set threshold. Limitations of this algorithm are that it does not usually converge to a global minimum and may not result in N clusters if zero elements are allocated to some of the clusters initially. Still, the algorithm is easy to implement and is effective if the data to be clustered is well separated. The complete algorithm is as follows:

1. Choose the number of clusters N and randomly choose N center points for these clusters.
2. Allocate each data element to the cluster whose center is nearest to the data element(for a 2D space the distance measure is Euclidean distance).
3. Update the clusters centers by finding the means of elements in each cluster.
4. Repeat steps 2 and 3 till the cluster centers do not change between two successive iterations.

Small variations can be made in the algorithm described above while implementation without changing the results in any significant way such as assigning the data points randomly to the clusters initially and finding their means or changing the convergence condition in step 4.

Clustering Space

First step in clustering feature points based on common motion is to define a clustering space that enables meaningful clustering of features. Consider two frames I_1 and I_2 of the sequence shown in Figure 1.2. Let the number of features tracked from frame I_1 to I_2 be n. If the position of a feature i in frame I_1 is (x_i, y_i), then its position in frame I_2 is given by $(x_i + dx_i, y_i + dy_i)$, where dx_i and dy_i is the distance moved by the feature in the x and y direction respectively. The total distance moved by the feature i in two frames is given by

$$d_i = \sqrt{dx_i^2 + dy_i^2}.$$

The clustering space is defined by the total displacement of the features and the frame difference, then for a pair of frames, the clustering space is unidimensional and the data set is given by $[d_1, d_2, \ldots, d_n]^T$. If the motion is present predominantly in one direction(as in the case of Figure 1.2 where motion is only in the horizontal direction), then the data set can be represented as $[dx_1, dx_2, \ldots, dx_n]^T$.

For the reasons described in Chapter , motion computed by considering only two successive frames may not be sufficient for correct segmentation of scene and hence, multiple frames need to be considered. Now, the clustering space becomes a 2D space such that each data point is represented by $D_i = [d_i \; k]^T$, where $i = 1, \ldots, n$ are the features and $k = 1, \ldots, F$ are the frames. The distance between each data point D_i in the clustering space is measured as the Euclidean distance.

Fitting Lines

Considering each point D_i separate while clustering is valid only if the displacement is calculated by considering only two frames i.e., for a 1D clustering space. When multiple frames are considered then the each data point D_i associated with a feature cannot be considered as a separate or distinct from the rest of the data points in a 2D clustering space. If clustering is performed with this assumption, then the results are not accurate as shown in Figure 3.1. This is because each object moving through the sequence has some

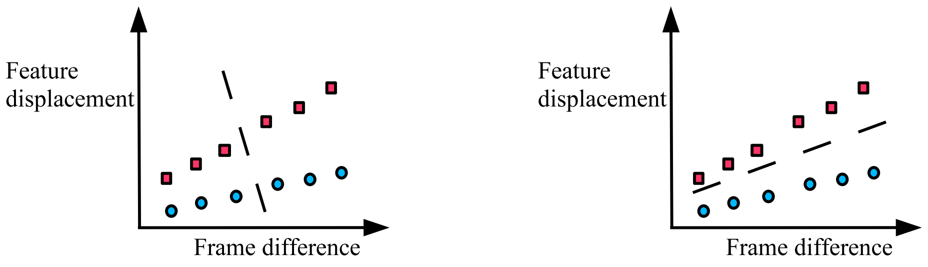

Figure 3.1 Plot of feature displacement belonging to two different objects over time . LEFT: Clustering using points produces incorrect clusters. RIGHT: Clustering by fitting lines correctly clusters two different motion trajectories.

features associated with itself and trajectories of these features are related to each other due to common motion. As seen from the Figure 3.1, there are two distinct tracks pertaining to the foreground and the background regions and since in this case the motion is linear, the tracks are straight lines in the clustering space. So instead of clustering individual data points, lines are fitted to the data which represent motion of different objects in the screen.

Line fitting using K-means is similar to the K-means algorithm described previously, with only a few variations. Instead of points belonging to clusters, now the data points define a line through the clustering space and instead of finding distance of a data point from the center of the cluster, distance of a data point from the lines is calculated. If $y = mx + r$ is the equation of the line, where m and r are the slope and the intercept respectively, then, the best line fitting a set of data given by (x_i, y_i) is given by

$$\begin{pmatrix} r \\ m \end{pmatrix} = \begin{pmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

Here n is the number of data points in the given set. After solving the above equation, the slope and the intercept is obtained as

$$m = \frac{n(\sum_{i=1}^{n} x_i y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2}$$

$$r = \frac{\sum_{i=1}^{n} y_i - m(\sum_{i=1}^{n} x_i)}{n}$$

The equation of a line in 2D space in parametric form is given by $ax + by + c = 0$. The distance from a point (x_i, y_i) to the line is given by

$$d = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}}.$$

If the slope is given by $m = -\frac{a}{b}$ and the intercept is $r = -\frac{c}{b}$, then the distance of a point (x_i, y_i) to the line is given by

$$d = \frac{|-mx_i + y_i - r|}{\sqrt{m^2+1}}.$$

The K-means algorithm, modified to fit lines to the given data is described below:

1. Choose the number of lines N and randomly choose N set of parameters (slope and intercept) for these lines.

2. Allocate each data element to the line closest to the data element.

3. Update the line parameters using the all points fitted by the line for N lines.

4. Repeat steps 2 and 3 till the line parameters do not change between two successive iterations.

Results of clustering by using the above modified K-means algorithm are shown in Figure 3.2. It can be seen from the plots of feature displacement with respect to frame difference that features belonging to the foreground regions have a lot of variations in the motion which leads to errors in clustering.

Clustering Using an Affine Motion Model

Clustering features in previous sections assumes a translational motion of the features. This assumption is well suited for sequences where there is linear motion of the objects (undergoing translation and rotations but no out of plane rotation) in the foreground such as in the Pepsi can sequence. Image motion can be modeled in a more realistic manner by using an affine motion model, that accounts for scaling and shear in addition to the translation and rotation.

The affine partitioning algorithm partitions the given data into distinct groups that adhere to affine motion models with different parameters. The data in the present context refers to the feature motion in the affine space. Let the total number of features be n and N be the labels associated with different affine models according to which the features are

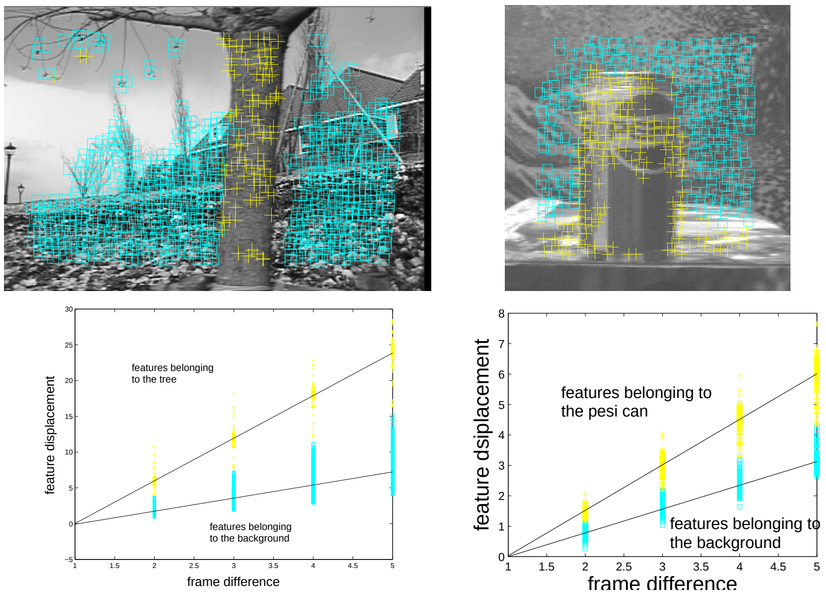

Figure 3.2 TOP: Results of clustering of feature motion for flower-garden sequence(left) and Pepsi can sequence(right). BOTTOM: Motion trajectories of the foreground and background regions in both the sequences .

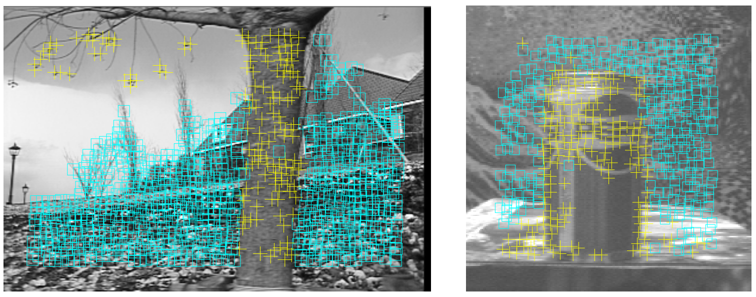

Figure 3.3 Clustering feature motion using an affine motion model. LEFT: The flowergarden sequence. RIGHT: The Pepsi can sequence.

to be grouped. $\mathbf{x}_1^{(j)}, \ldots, \mathbf{x}_n^{(j)}$ be the coordinates of the features in the j^{th} frame. Using an affine morion model, the coordinates of the i^{th} feature in the $(j+1)^{th}$ frame are given by

$$\mathbf{x}_i^{(j+1)} = A\mathbf{x}_i^{(j)} + B$$

where,

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and } B = \begin{pmatrix} t_x \\ t_y \end{pmatrix}.$$

The parameters of the N different affine motion models are given by $(A_1, B_1), \ldots, (A_N, B_N)$ and each feature is assigned a label $\phi_k,\ k = 1, \ldots, N$ depending upon the affine model to which the feature belongs. The feature belongs to the affine model for which it gives the least residue. Residue of the i^{th} feature for the k^{th} affine model is given by

$$r_{(i)k} = \|A_k\mathbf{x}_i^{(j)} + B_k - \mathbf{x}_i^{(j+1)}\|.$$

Based on the residue, the label for each feature is assigned as $\phi = k$, such that k is the index of $min(r_{(i)k}; k = 1, \ldots, N)$. The process of assigning labels to the features and recomputing the affine models based on the assigned labels is carried out iteratively till the feature labels do not change significantly through the two successive iterations or the average residue of the whole feature groups fall below a certain preset threshold. The algorithm to partition the feature motion in two affine groups is described below:

```
1. Initialize the two sets of affine parameters $(A_1, B_1)$ and $(A_2, B_2)$.
2. While the labels change
   (a) for $i = 1$to $n$
       i. $r_{(i)1} = \|A_1\mathbf{x}_i^{(j)} + B_1 - \mathbf{x}_i^{(j+1)}\|$
       ii. $r_{(i)2} = \|A_2\mathbf{x}_i^{(j)} + B_2 - \mathbf{x}_i^{(j+1)}\|$
       iii. if $r_{(i)1} < r_{(i)2}$
                $\phi_i = 1$
            else
                $\phi_i = 2$
```

(b) Recompute (A_1, B_1) and (A_2, B_2) using least squares.

The clustering using affine motion model is shown in Figure 3.3. It can be noted that the tree and the Pepsi can are segmented in a better manner as compared to clustering using just translation of features(see Figure 3.2).

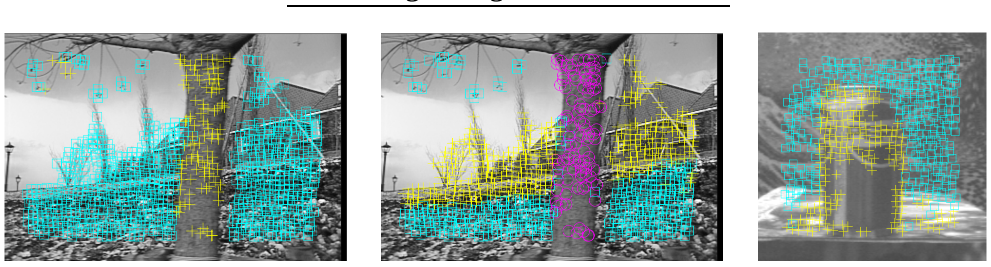

Figure 3.4 Results of clustering using normalized cuts. LEFT: The flower-garden sequence with 2 clusters. CENTER: the flower-garden sequence 3 clusters. RIGHT: The Pepsi-can sequence with 2 clusters

Normalized cuts algorithm proposed in [12] views the segmentation as a graph partitioning problem. Let $G = (V, E)$ be the graph whose nodes are the points to be clustered and let $w(i, j)$ be the weight of the edge between nodes i and j representing the similarity between the nodes. The objective is to partition the graph into m sets of nodes.

A *cut* is defined as the sum of weights of the edges that are removed while partitioning the graph into two parts. For partitioning the graph $G = (V, E)$ into sets of nodes A and B, the cut is given by

$$cut(A, B) = \sum_{u \in a, v \in B} w(u, v)$$

Minimizing the $cut(A, B)$ may produce the desired sets A and B but in many cases applying minimum cut is not a feasible idea because it leads to cutting of small isolated nodes. The normalized cuts algorithm, normalizes the cut so as to produce an optimum partition that is not biased in favor of cutting small parts of the graph. Normalized cuts, or Ncuts, is defined in terms of *cut* and *association*. Association of a region with the whole set of nodes is defined as the sum of weights of the edges between the nodes in the region and all the nodes in the graph. Hence, association of region A with the complete set V is given by

$$asso(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

The normalized cut for a graph is given by

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)}$$

If there are N nodes in the graph then a N dimensional vector d is defined such that

$$d(i) = \sum_{j=1}^{N} w(i, j)$$

where, i and j are the nodes of the graph. In present case, the nodes are the tracked features in the image, the edge weight $w(i, j)$ is the feature motion between two frames and $d(i)$ gives the similarity between the motion of i^{th} feature and all the other features. An $N \times N$ diagonal matrix D is defined with d on its diagonal, while another $N \times N$ symmetrical matrix W is constructed such that $W(i, j) = w(i, j)$. Let x be the N partitioning vector that indicates whether a node belongs to the set A or set B ans is given by

$$x = \begin{cases} 1 & \text{if node } i \text{ is in } A; \\ -1 & \text{otherwise.} \end{cases}$$

Finding a vector x that shows the partitions of the nodes is a NP Hard problem and hence, Ncuts solves the problem in continuous domain. A continuous approximation of x is given by

$$y = (1 + x) - \frac{\sum_{x_i > 0} d_i}{\sum_{x_i < 0} d_i}(1 - x)$$

The eigenvector associated with the second smallest eigenvalue of the system

$$(D - W)y = \lambda D y$$

outputs the optimum partition of the graph. To further sub-divide the two sets, the Ncuts algorithm is applied to the set of nodes till the value of the cut is below a pre-specified threshold. The results of using normalized cuts algorithm for clustering feature motion on two sequences are shown in Figure 3.4. The algorithm is sensitive to the cut threshold and a small variation in the threshold may produce significantly different results.

CHAPTER 4
SEGMENTATION BY REGION GROWING

This chapter describes a region growing algorithm to group features based on their motion in two frames. The feature grouping algorithm presented in this chapter approaches the problem from a different point of view as compared to the one described in Section , such that instead of starting with all the features as a single cluster of data and dividing it, individual features are grouped together and the group is grown progressively. The principal idea behind both the approaches remains the same, i.e., formation of feature groups based on the motion of features. The next chapter describes methods to use the algorithm presented here as the basis to perform motion segmentation over multiple frames.

Grouping Features Using Two Frames

Once features are tracked from one image frame to another, the features are grouped using an affine motion model on the displacements of the coordinates of the features. In other words, whether a feature f is incorporated into a group with an affine motion model A is determined by measuring the difference

$$\texttt{Diff}(f, A) = ||A\mathbf{x}^{(ref)} - \mathbf{x}^{(curr)}||,$$

where $\mathbf{x}^{(curr)}$ and $\mathbf{x}^{(ref)}$ are the coordinates of the feature in the current and reference frames, respectively, and $||\cdot||$ is the L_2 norm. For simplicity, homogeneous coordinates are used in this equation, so that A is a 3×3 matrix with $[\,0 \quad 0 \quad 1\,]$ as the bottom row.

A region growing approach is adopted, as shown in the algorithm `GroupFeatures`. First, all the features are labeled as ‘ungrouped’, and a feature point is selected at random as the seed point to begin a new group $\mathcal{F}$. The motion of the group is computed by fitting affine parameters to the motion of the feature and all of its immediate ungrouped neighbors $\mathcal{N}_u(f)$ The process continues to add any neighbor of a feature in the group whose motion is similar to the motion of the group. The function $\mathcal{S} \leftarrow \texttt{Similar}(\mathcal{N}_u(\mathcal{F}), A, \tau)$ returns all the ungrouped neighbors f of the features in $\mathcal{F}$ for which $\texttt{Diff}(f, A) < \tau$, where τ is

Algorithm: `GroupFeatures`

Input: A set of features with motion vectors
Output: A set of groups of features

```
1. Set all features to 'ungrouped'
2. While at least one feature is 'ungrouped',
   (a) Select a random 'ungrouped' feature $f$
   (b) Set $\mathcal{F} \leftarrow \{f\} \bigcup \mathcal{N}_u(f)$
   (c) Compute affine motion model $A$ of $\mathcal{F}$
   (d) Repeat until $\mathcal{F}$ does not change
       i. Set $\mathcal{F} \leftarrow \{f'\}$, where $f'$ is the feature closest to the centroid of $\mathcal{F}$
      ii. Repeat until $\mathcal{S}$ is empty
          (a) Find similar nearby features
              $\mathcal{S} \leftarrow \texttt{Similar}(\mathcal{N}_u(\mathcal{F}), A, \tau)$
          (b) Set $\mathcal{F} \leftarrow \mathcal{F} \bigcup \mathcal{S}$
          (c) Compute affine motion model $A$ of $\mathcal{F}$
   (e) Set all features in $\mathcal{F}$ to 'grouped'
```

a threshold indicating the maximum motion difference allowed. When no more features can be added to the group, the group is reset to the feature closest to the centroid of the group, and the process begins again. Convergence is usually obtained within two or three iterations.

Now that a single group has been found, all the features in the group are labeled with a unique group id. The procedure then starts again using another random feature as the seed point among all those that have not yet been grouped, and the process is repeated until all features have been grouped. Note that the algorithm automatically determines the number of groups. Figure 4.1 shows the iterative growing of a feature group from a single seed point. As it is seen from the images, the it takes only about 4 iterations for the feature group to achieve a stable size.

Finding Neighbors

Finding neighborhood features is an essential step for growing the feature groups. A straightforward way is to use a spatial search window to search the neighboring features and

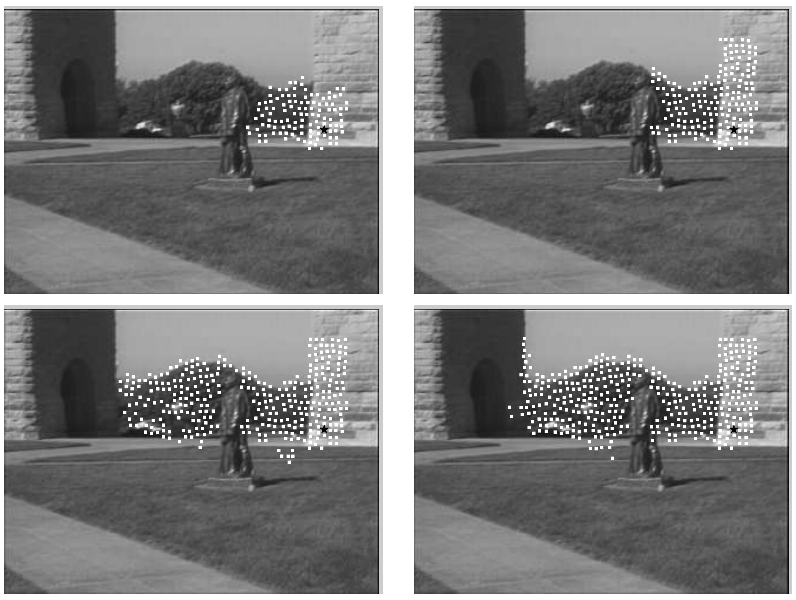

Figure 4.1 Formation of a feature group iteratively from a single seed point by fitting the affine motion model to the neighborhood features. The green star indicates the seed point.

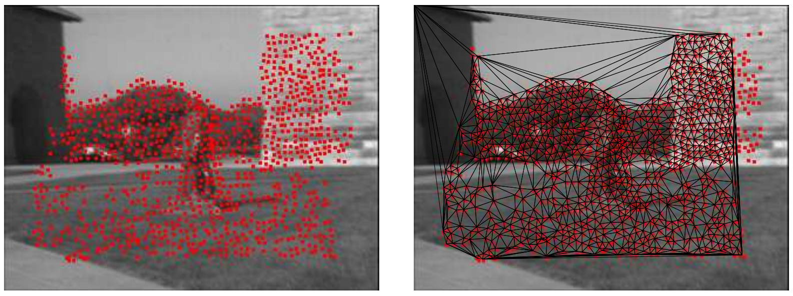

Figure 4.2 Delaunay triangulation for finding neighbors of a feature point in the image. LEFT: Feature points overlaid on an image frame. RIGHT: Delaunay triangulation connects every fetaure point with its neighbors such that each circumcircle thus formed does not contain any features.

fit an affine model to the selected features. Another approach is to establish neighborhood criterion using Delaunay triangulation. In the context of a 2D plane, Delaunay triangulation refers to the process of sub-dividing it into triangular regions. For given set of points in a 2D plane, Delaunay triangulation gives the lines that join a point with its immediate neighbors such that the circumcircle formed by a triangle does not consist of any points (see Figure 4.2). Delaunay triangulation is a dual of Voronoi diagram which can be obtained by drawing lines bisecting the edges of the Delaunay triangles. The advantage of using Delaunay triangulation to find the neighbors is that it leads to efficient search techniques for finding neighbors.

CHAPTER 5
CONSISTENCY OF FEATURE GROUPS

Like previous algorithms, the grouping procedure of the last section operates on exactly two image frames. If the frames are spaced closely together, then slowly moving objects will not be detected. On the other hand, if the frames are spaced far apart, then the affine motion assumption and the feature tracking are likely to fail. As a result, algorithms that use a constant the inter-frame spacing (whether operating on a pair of frames or on a spatiotemporal block of frames) make potentially dangerous assumptions about the amount of motion of the objects in the scene. This chapter describes a procedure to select consistent feature groups and an incremental segmentation algorithm that maintains these consistent feature groups over time.

Finding Consistent Groups

Three important parameters that affect the feature grouping via region growing algorithm described in previous section are the following: the pair of frames for which the feature motion is considered (the starting frame and ending frame), the grouping error threshold, and the seed point for the algorithm. Of these three parameters, the seed point is chosen at random from the available features. If the starting frame and the ending frames as well as the grouping error threshold are kept constant then the seed feature point for the algorithm is the only variable parameter than can produce different feature grouping results. This means, with random seed points, the features are grouped differently every time the region growing algorithm is applied to the same set of images with the same grouping error threshold as shown in Figure 5.1. But if the regions indeed belong to a single object in the scene, whose motion can be defined by a motion model, then changing the seed points should not affect the grouping of features of that object. This forms the main idea behind the feature group consistency criterion.

The feature group consistency criterion can be used to extend the two frame region growing algorithm to work over a long sequence by accumulating the feature motion for more than two frames. Since all the consistent feature groups are not precipitated by considering the motion of feature points in two frames, evidence accumulation is done and initial segmentation of the scene is achieved when sufficient evidence is accumulated. Once the scene is segmented, individual feature groups are tracked and their motion model is updated as the sequence progresses to include new features appearing in the scene.

In the algorithm `GroupConsistentFeatures`, we start with a pair of frames, retaining only those features that are successfully tracked through the two frames. N_s seed points are randomly selected from the image and the `GroupFeatures` algorithm described in section is performed using all N_s seed points. If the number of features present in the scene are m then a consistency matrix $c_{m \times m}$ is constructed, with each row corresponding to a feature while every column entry corresponding to the feature grouped along with it and it is updated with the results of `GroupFeatures` algorithm for each seed point. If f and f' are the features then c is updated as

$$c(f,f') = \begin{cases} c(f,f')+1 & \text{if } f \text{ and } f' \text{ belong to the same group;} \\ c(f,f') & \text{otherwise.} \end{cases}$$

A set of features is said to form a consistent group if $c(f,f') = N_s$ for all features in the set. The collection $\mathcal{C}$ of consistent groups larger than a minimum size $n_{\min}$ are retained, and all the remaining features are set again to ‘ungrouped’. Figure 5.1 displays the varying groups for different seed points on an example sequence, along with the consistent groups. The `GroupFeatures` algorithm is applied all over again with N_s number of seed points to the ungrouped features. But now, the frame window for the feature motion is increased while keeping the reference frame constant. There are some features that are not grouped consistently or even if they are, they form very small groups and hence, are ignored at the moment only to be considered in future to be assigned to any other existing feature group. This is done in order to prevent formation of small groups during the initial segmentation which may lead to over-segmentation.

Maintaining Feature Groups Over Time

After the initial scene segmentation, the feature groups are tracked through the sequence while updating the affine motion model for each feature group. This process is important

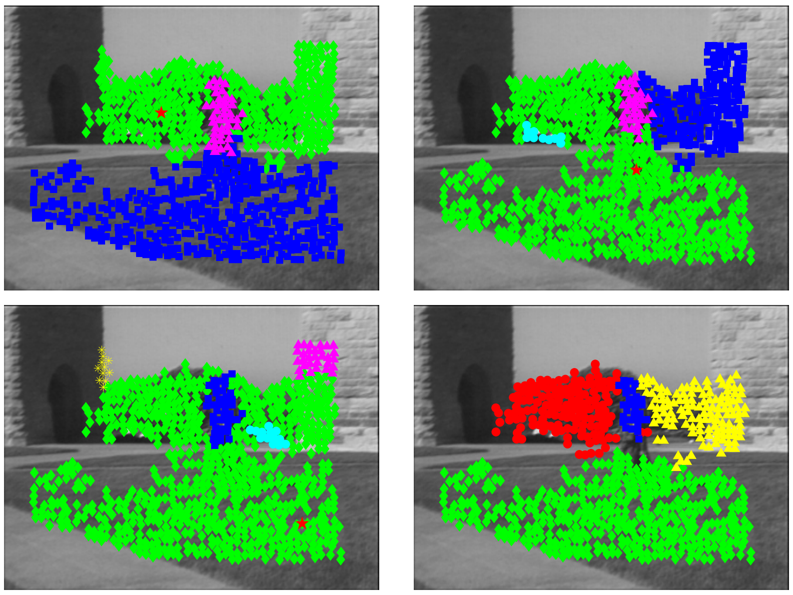

Figure 5.1 Results of region growing algorithm with different seed points for frames 1 to 7 and grouping threshold δ_g = 1.5. The consistent feature groups are those which remain unchanged even if the seed points are changed and are shown in the last image (bottom-right).

Algorithm: `GroupConsistentFeatures`

Input: A set of features with motion vectors
Output: A set of groups of features, and a set of ungrouped features

```
1. $c(f, f') \leftarrow 0$ for every pair of features $f$ and $f'$
2. for $i \leftarrow 1$ to $N_s$,
    (a) Run GroupFeatures
    (b) For each pair of features $f$ and $f'$, increment $c(f, f')$ if $f$ and $f'$ belong to the
        same group
3. Set $\mathcal{C} \leftarrow \{\}$     (empty set)
4. Repeat until all features have been considered,
    (a) Gather a maximal set $\mathcal{F}$ of consistent features such that $c(f, f') = N_s$ for all
        pairs of features in the set
    (b) if $|\mathcal{F}| > n_{\min}$, then $\mathcal{C} \leftarrow \mathcal{C} \bigcup \mathcal{F}$
5. Set all features that are not in a large consistent feature set (i.e., there is no $\mathcal{F} \in \mathcal{C}$
   such that $f \in \mathcal{F}$) to 'ungrouped'
```

for two reasons. First, the initial segmentation of the scene may not be perfect because two different image regions can be grouped as one due to lack of sufficient evidence. So the group may be required to split into two or more than two groups further down the sequence. Second, as the sequence progresses, some features are lost due to occlusion or changes in the scene while new features are added which need to be included in the feature groups to make the algorithm work over a long sequence.

A challenging situation arises when a new object enters the scene and needs to be segmented reliably. A straightforward approach would be to consider only those features that do not belong to any of the feature groups at the given instant of time, and group them according to the feature consistency criterion. This approach would require us to look for the frames where the number of features that do not belong to any group is high as these frames have higher probability of segmenting the new object. However, due to camera jitter and motion blur, large number of features are lost between consecutive frames and subsequently replaced during the course of the sequence. As a result, unusually high number of new features (features not assigned to any of the feature groups) are obtained in some frames which do not coincide with the appearance of a new object in the scene.

As shown in the algorithm `MaintainGroups`, our approach performs three computations when a new image frame becomes available. First, the consistent grouping procedure just described is applied to all the ungrouped features. This step generates additional groups if sufficient evidence for their existence has become available.

Secondly, the consistent grouping procedure is applied to the features of each existing group. If a group exhibits multiple motions, then it will be split into multiple groups and/or some of its features will be discarded from the group and labeled instead as ‘ungrouped’. Because groups tend to be stable over time, we have found that this procedure does not need to be performed every frame. To save computation, we apply the procedure only when fewer than 50% of the features in a group are at least as old as the reference frame for the group, and we reset the reference frame after running the procedure.

The third computation is to assimilate ungrouped features into existing groups. For each ungrouped feature f, we consider its immediate neighbors $\mathcal{N}_g$ (in the Delaunay triangulation) that are already grouped. If there are no such neighbors, then no further tests

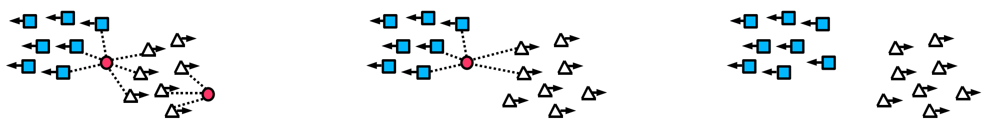

Figure 5.2 Addition of new features to existing feature groups. The squares and the trinagles represent different feature groups with the arrows showing the motion of the features while the circles are the newly added features. The dotted lines represent the neighbors of the newly added features.

are performed. If there is exactly one such neighbor, then the feature is assimilated into the neighbor's group if the motion of the feature is similar to that of the group, using the same threshold τ used in the grouping procedure. If there is more than one such neighbor belonging to different groups, then the feature is assimilated into one of the groups only if its motion is similar to that of the group and is dissimilar to that of the other groups, using the threshold τ.

Comparing motion of newly added features with all the neighboring feature groups ensures that the feature is not accidently added to a wrong group. This means the feature remains ungrouped for a period of time till its motion is ascertained to be similar with one of the existing feature group and dissimilar to the rest. During this period many new features are added and the existing features are lost from the existing feature groups making the computation of the affine motion parameters for a feature group a challenging task. Motion of the candidate feature can be compared with the existing feature groups in multiple ways. One method would be to normalize the motion of the candidate feature over number of frames for which its motion is being considered in the sequence and then compare it with the existing feature groups. An alternative approach is followed that calculates the motion parameters of an existing feature group by considering only those features that are successfully tracked through the frames in which the candidate feature has been present. This leads to a greater flexibility while calculating the motion parameters of the feature groups and performs well in situations where there is a lot of non-uniform motion and motion blur. Addition of new features to the existing feature groups is illustrated in Figure 5.2.

Algorithm: `MaintainGroups`

Input: A set of groups, and a set of ungrouped features
Output: A set of groups, and a set of ungrouped features

```
1. Grouping. Run GroupConsistentFeatures on all the ungrouped features
2. Splitting. For each group $\mathcal{F}$,
   (a) Run GroupConsistentFeatures on the features in $\mathcal{F}$
3. Assimilation. For each ‘ungrouped’ feature $f$,
   (a) $\mathcal{S} \leftarrow \mathcal{N}_g(f)$
   (b) If $\mathcal{S}$ is nonempty,
       i. Set $\mathcal{G}$ to the set of groups to which the features in $\mathcal{S}$ belong
       ii. For each $\mathcal{F} \in \mathcal{G}$,
           (a) Compute affine motion model $A$ of $\mathcal{F}$
           (b) Set $\mathcal{F} \leftarrow \mathcal{F} \bigcup \{f\}$ and $f$ to ‘grouped’ if
               • $\mathtt{Diff}(f, A) \leq \tau$ and
               • $\mathtt{Diff}(f, A) \leq \mathtt{Diff}(f, A') + \tau$ for the affine motion $A'$ of any other
                 group $\mathcal{F}' \in \mathcal{G}$
```

CHAPTER 6
EXPERIMENTAL RESULTS

The algorithm was tested on four grayscale image sequences shown in Figure 6.1. The first sequence is 20 frames long and shows a basketball player in motion. The player is successfully segmented from the background as shown in Figure 6.2. The second sequence is the 101 frame long mobile-calender sequence in which a toy train and a ball move in the foreground in a horizontal direction, while a calender in the behind the train moves in the vertical direction relative to the stationary background as the camera zooms out. The results of segmentation for this sequence are shown in Figure 6.3. The train and the ball are segmented within 3 frames while the calender takes 7 frames to separate from the background. The third sequence is of 35 frames and shows a car moving behind a map and reappearing from the other side. The car is segmented in the second frame while the building in the background is segmented in frame the map and the ground are segmented in frame 5. The map and the ground are grouped in frame 8. The results for this sequence are shown in Figure 6.4.

The statue sequence is the longest and the most challenging of all as it is shot by a hand held camera and is marred by very high motion blur, high inter-frame motion and the objects appearing and being occluded and moving with a non-uniform motion. The results of the algorithm for this sequence are shown in Figure 6.6. By frame 8, four consistent feature groups belonging to the statue, the wall, the grass and trees are formed. The bicyclist enters in frame 151 (detected in frame 185), becomes occluded by the statue in frame 312, and emerges on the other side of the statue in frame 356 (detected again in frame 444). Because the algorithm does not attempt correspondence between occluded and disoccluded objects, the bicylist group receives a different group id after the disocclusion. The pedestrian enters the scene in frame 444 and is segmented successfully, although the non-rigid motion prevents the feature tracker from maintaining a large number of features throughout. The pedestrian occludes the statue from frames 346 to 501, after which the statue is regrouped. Features on the grass are not regrouped due to an apparent error in the KLT feature tracker

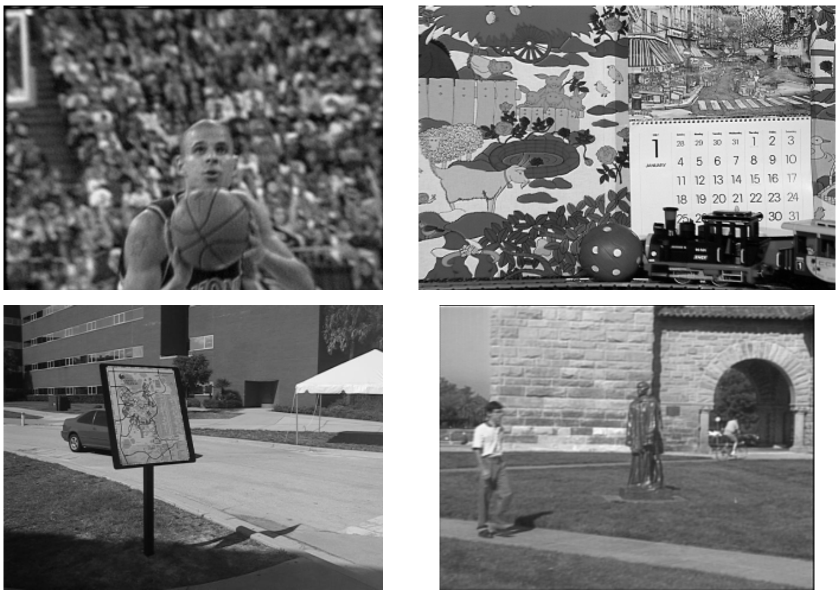

Figure 6.1 One frame from each of the four sequences used for testing the proposed algorithm;(LEFT TO RIGHT) the basketball sequence, the mobile-calendar sequence, the car-map sequence and the statue sequence.

that prevents features from being replenished on the grass after the pedestrian passes.

Figure 6.5 shows the plots for number of feature groups at each frame for all the four sequences. In many occasions the object appears in the scene long before it is segmented. This is especially true in the case of the statue sequence. One reason for this is the fact that due to motion blur and high inter-frame motion, the feature tracking is not perfect and hence, the objects keep losing features as in the case of the pedestrian in the statue sequence. Amount of texture and the contrast between the background and the foreground also leads to the apparent delay in the segmentation of the bicyclist in the statue sequence.

For all the above sequences, 1000 features were detected and tracked, replacing the lost features along the sequence. The grouping error threshold τ is set at 1.5. The value of τ is not of very high significance as changing it only changes the time required by different objects for segmentation. This is shown in Figure 6.7 where, τ is reduced to 0.7 and consequently the segmentation of the statue takes place at frame 4 instead of 7 as shown in Figure 6.6. Similarly when the algorithm runs on a faster statue sequence generated by dropping every alternate frame with the same value of τ (1.5), similar results are produced in twice as fast time. The results for this experiment are shown in Figure 6.8.

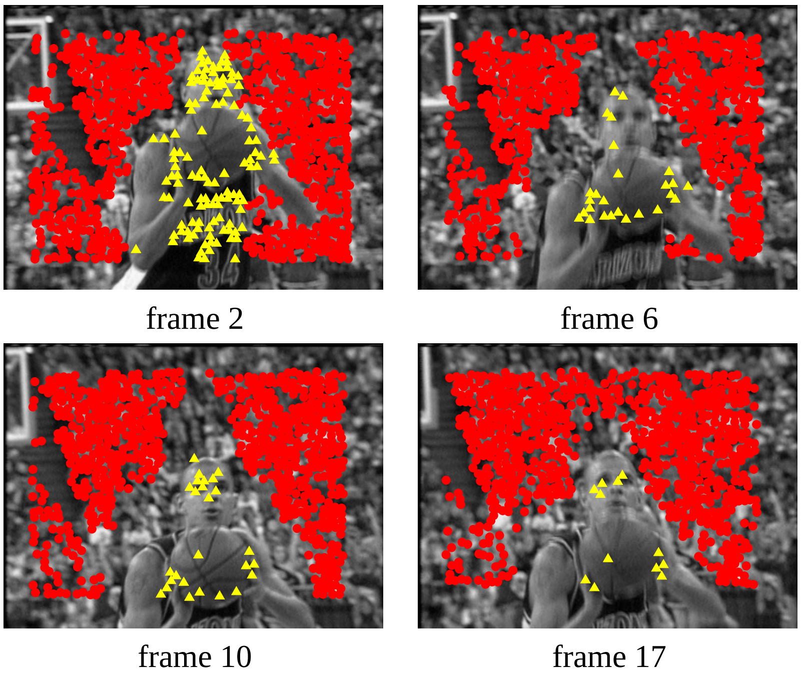

Figure 6.2 Results of feature grouping algorithm on a 20 frame basketball sequence, with different group is shown by features of different color and shape.

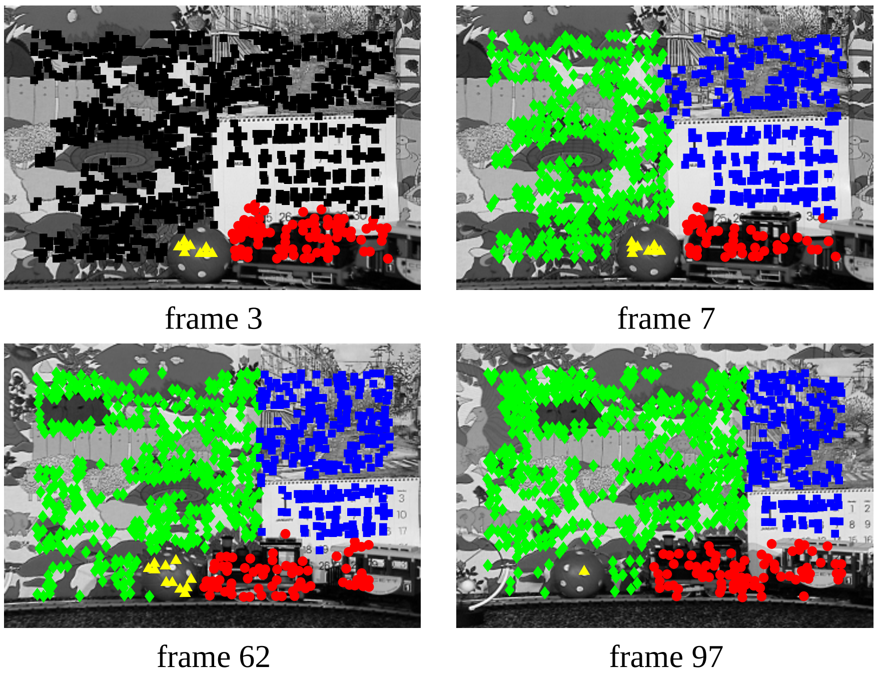

Figure 6.3 Results of feature grouping algorithm on a 101 frame mobile-calendar sequence, with different group is shown by features of different color and shape.

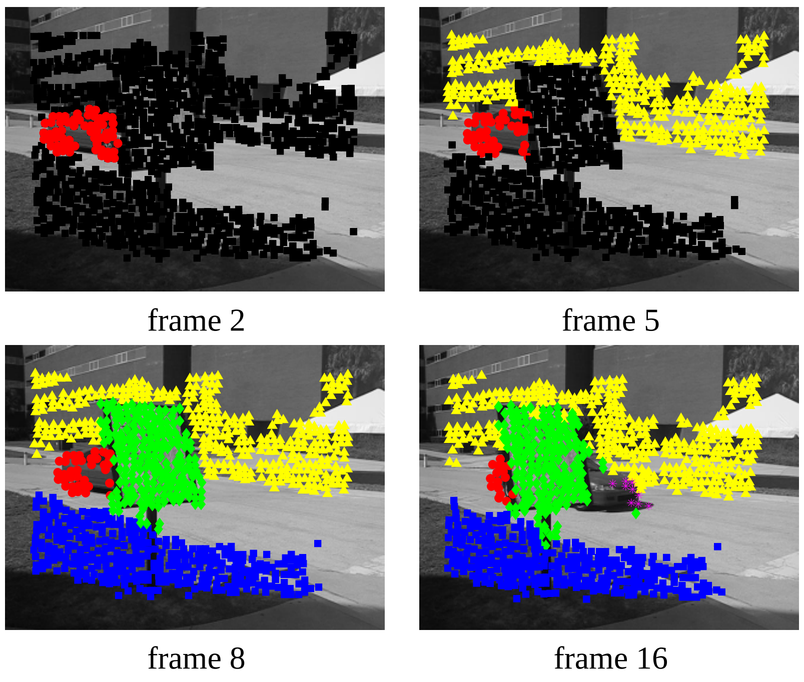

Figure 6.4 Results of feature grouping algorithm on car-map sequence, with different group is shown by features of different color and shape.

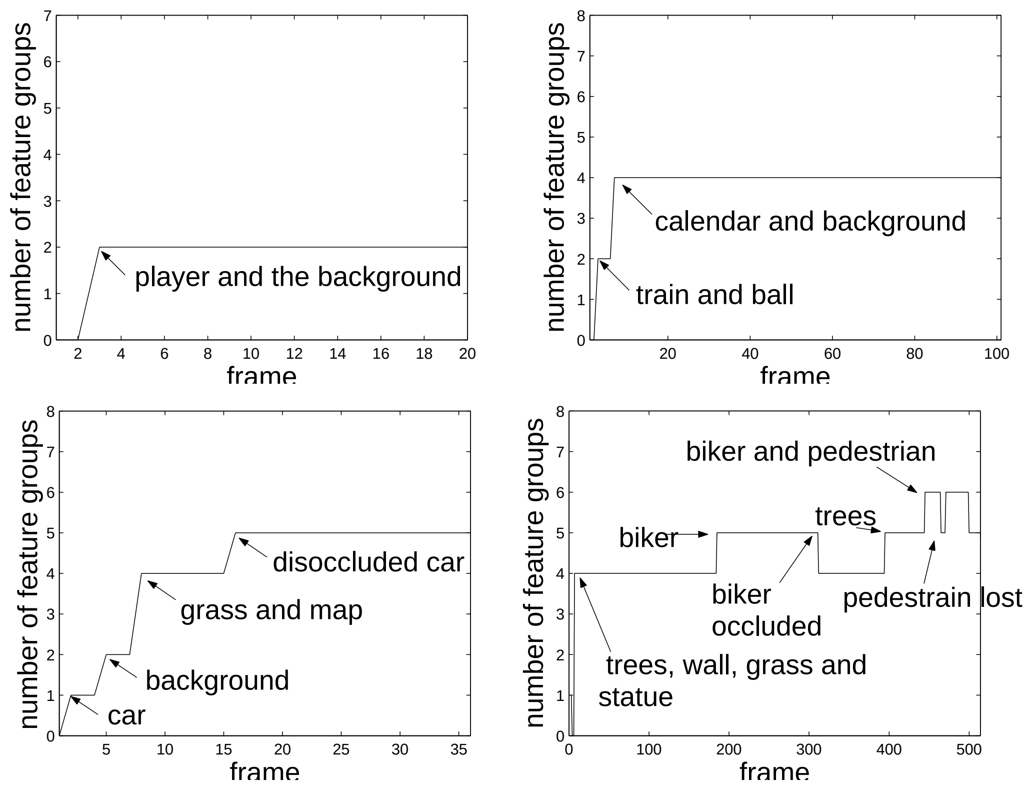

Figure 6.5 The dynamic progress of the algorithm on the four sequences, plotted as the number of feature groups versus time. The number of groups is determined dynamically and automatically by the algorithm.

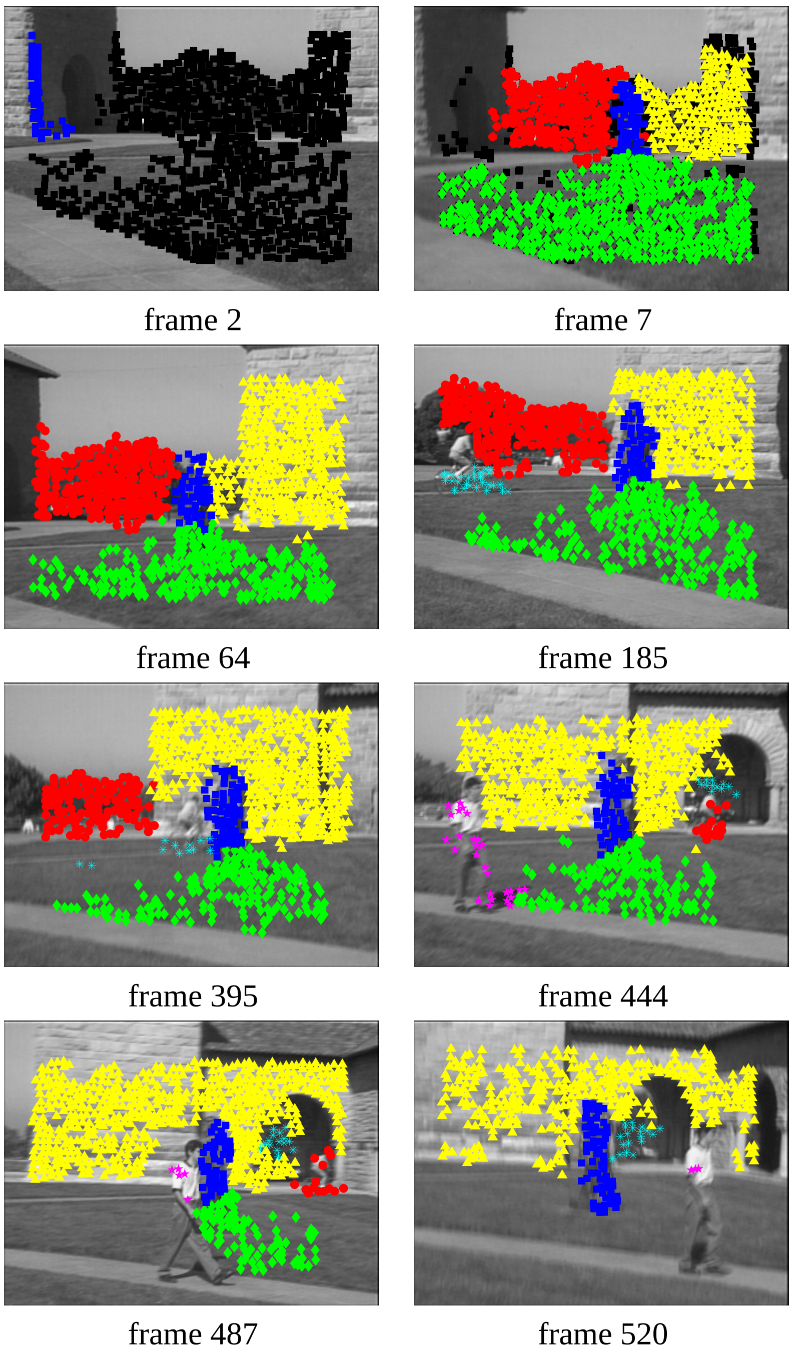

Figure 6.6 Results of feature grouping algorithm on a 520 frame statue sequence. The feature points in black in the first two frames are ‘ungrouped’. After the first frame in which all of the features are ungrouped, subsequent images show the formation of new groups or the splitting of existing groups due to the arrival or departure of entities in the scene.

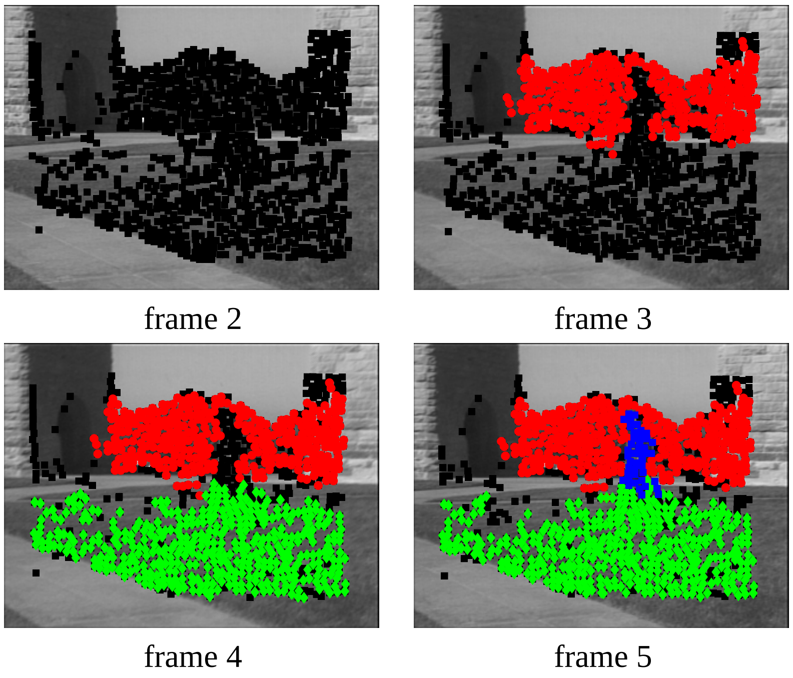

Figure 6.7 Results on the statue sequence with $\tau = 0.7$. By reducing the threshold by a factor of two, the segmentation occurs in nearly half the time.

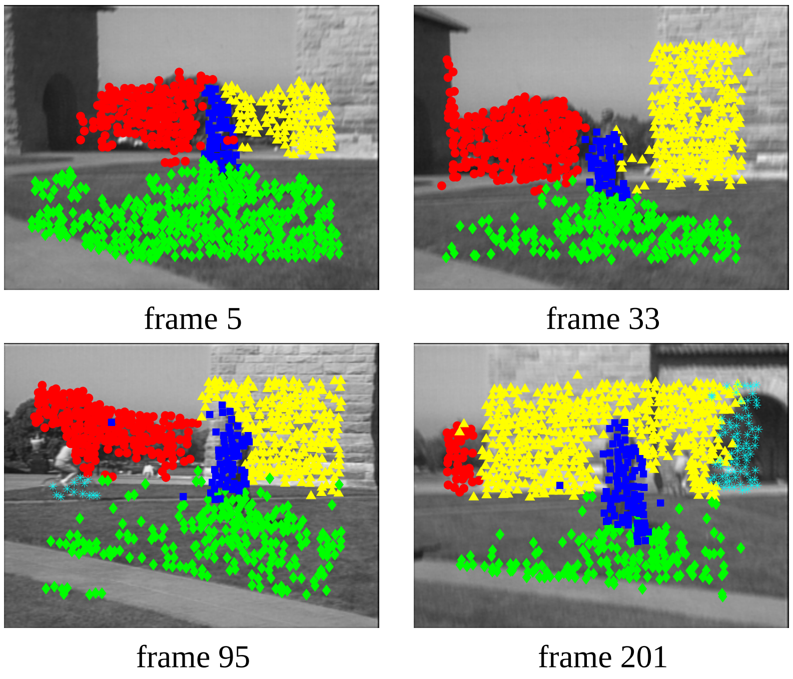

Figure 6.8 Results of feature grouping algorithm on a 260 frame fast statue sequence generated by collecting every alternate frame from the original sequence. The frame numbers below the figures correspond to those of the original statue sequence of Figure 6.6

CHAPTER 7
CONCLUSION

The paper presents the concept of motion segmentation from a novel point of view in which the evidence accumulated over a period of time is key to perform motion segmentation. The evidence for segmentation in this case is the feature motion and sufficiency of evidence is tested using a feature group consistency criterion. The approach is unique as it is based on a feature grouping algorithm that works on feature motion between two frames but at the same time the results of this two frame motion segmentation step can be effectively used to segment the scene over time and track the feature groups over a long sequence by constantly adding new features to the existing groups and updating the motion model of the groups. The result demonstrated on numerous challenging sequences show the success of the algorithm. The results of the algorithm on faster sequence and changing grouping threshold are comparable and are on expected lines.

The present algorithm is designed to handle only those features that are tracked successfully between two frames. One aspect of the algorithm that needs improvement is handling of feature motion, independent of the success of feature tracking,to produce accurate segmentation results. Future work also involves handling the occlusion and dis-occlusion of the objects in the scene by a possible use of a layered motion model, precise assignment of features to the feature groups and increasing the overall efficiency of the algorithm.

BIBLIOGRAPHY

[1] S. Ayer and H. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *Proceedings of the International Conference on Computer Vision*, pages 777–784, 1995.

[2] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real time computer vision system for measuring traffic parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[3] Stanley T. Birchfield. *Depth and Motion Discontinuities*. PhD thesis, Dept. of Electrical Engineering, Stanford University, June 1999.

[4] Michael J. Black and David J. Fleet. Probabilistic detection and tracking of motion discontinuities. *Proceedings of the International Conference on Computer Vision*, pages 551–558, 1999.

[5] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the International Conference on Computer Vision*, pages 1071–1076, 1995.

[6] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. New Jersey: Prentice Hall, first edition, 2003.

[7] J. Gonzalez, I. Lim, P. Fua, and D. Thalmann. Robust tracking and segmentation of human motion in an image sequence. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[8] N. K. Kanhere, S. J. Pundlik, and S. T. Birchfield. Vehicle segmentation and tracking from a low angle off-axis camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[9] Q. Ke and T. Kanade. A robust subspace approach to layer extraction. In *IEEE Workshop on Motion and Video Computation*, pages 37–43, 2002.

[10] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages II:746–751, 2001.

[11] Oscar Nestares and David J. Fleet. Probabilistic tracking of motion boundaries with spatio-temporal predictions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:358–365, 2001.

[12] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997.

[13] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[14] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[15] John Y. A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, September 1994.

[16] Yair Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 520–526, 1997.

[17] Yair Weiss and Edward H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[18] J. Wills, S. Agarwal, and S. Belongie. What went where. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages I: 37–44, 2003.

[19] Jiangjian Xiao and Mubarak Shah. Accurate motion layer segmentation and matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.