

OBJECT TRACKING AND ACTIVITY RECOGNITION IN VIDEO ACQUIRED
USING MOBILE CAMERAS

by

ALPER YILMAZ

B.S. Yildiz Technical University
M.E. Istanbul Technical University
M.Sc. University of Central Florida

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2004

Major Professor:
Mubarak Shah

UMI Number: 3163642



UMI Microform 3163642

Copyright 2005 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2004 by Alper Yilmaz

ABSTRACT

Due to increasing demand on deployable surveillance systems in recent years, object tracking and activity recognition are receiving considerable attention in the research community. This thesis contributes to both the tracking and the activity recognition components of a surveillance system. In particular, for the tracking component, we propose two different approaches for tracking objects in video acquired by mobile cameras, each of which uses a different object shape representation. The first approach tracks the centroids of the objects in Forward Looking Infrared Imagery (FLIR) and is suitable for tracking objects that appear small in airborne video. The second approach tracks the complete contours of the objects, and is suitable for higher level vision problems, such as activity recognition, identification and classification. Using the contours tracked by the contour tracker, we propose a novel representation, called the action sketch, for recognizing human activities.

Object Tracking in Airborne Imagery: Images obtained from an airborne vehicle generally appear small and can be represented by geometric shapes such as circle or rectangle. After detecting the object position in the first frame, the proposed object tracker models the intensity and the local standard deviation of the object region defined by the shape model. It then tracks the objects by computing the mean-shift vector that minimizes the distance

between the kernel distribution for the hypothesized object and its prior. In cases when the ego-motion of the sensor causes the object to move more than the operational limits of the tracking module, a multi-resolution global motion compensation using the Gabor responses of consecutive frames is performed. The experiments performed on the AMCOM FLIR data set show the robustness of the proposed method, which combines automatic model update and global motion compensation into one framework.

Contour Tracker: Contour tracking is performed by evolving an initial contour toward the correct object boundaries based on discriminant analysis, which is formulated as a variational calculus problem. Once the contour is initialized, the method generates an online shape model for the object along with the color and the texture priors for both the object and the background regions. A priori texture and color PDFs of the regions are then fused based on the discrimination properties of the features between the object and the background models. The models are then used to compute the posteriori contour likelihood and the evolution is obtained by the Maximum a Posteriori Estimation process, which updates the contour in the gradient ascent direction of the proposed energy functional. During occlusion, the online shape model is used to complete the missing object region. The proposed energy functional unifies commonly used boundary and region based contour approaches into a single framework through a support region defined around the hypothesized object contour. We tested the robustness of the proposed contour tracker using several real sequences and have verified qualitatively that the contours of the objects are perfectly tracked.

Behavior Analysis: We propose a novel approach to represent human actions by modeling the dynamics (motion) and the structure (shape) of the objects in video. Both the motion and the shape are modeled using a compact representation, which is called the “action sketch”. An action sketch is a view invariant representation obtained by analyzing important changes that occur during the motion of the objects. When an actor performs an action in 3D, the points on the actor generate space-time trajectories in four dimensions (x, y, z, t) . Projection of the world to the imaging coordinates converts the space-time trajectories into the spatio-temporal trajectories in three dimensions (x, y, t) . A set of spatio-temporal trajectories constitute a 3D volume, which we call an “action volume”. This volume can be treated as a 3D object in the (x, y, t) space. The action sketch is generated from the action volume by analyzing the differential geometric surface properties, such as peaks, pits, valleys and ridges. These properties reflect the changes in the speed, the motion direction and the shape of the performing actor. We perform action recognition by computing a view invariant distance measure between the sketch generated from the input video and the set of known sketches in the database. Experimental results are provided for twenty eight actions.

Dedicated to my cute little daughter *Irem*, my wonderful wife *Selen*, and my caring parents

Aynur and Niyazi Yilmaz.

ACKNOWLEDGMENTS

I would like to thank to my thesis adviser *Mubarak Shah* for his valuable guidance and encouragement. He has not only been an adviser but a family to me during our tough times.

I would like to thank *Prof. Charles Hughes*, *Prof. Xin Li*, and *Prof. Niels da Vitoria Lobo* for serving as my committee members and for their valuable comments and suggestions.

Special thanks to *Lisa Spencer* for proof reading the complete thesis, as well as her English writing classes which made this thesis readable.

I would like to specially thank to my wonderful wife *Selen* for her patience and her encouragements. I believe she owns a big part of my success. In my memory are always my mother *Aynur* and my father *Niyazi*, to whom I dedicate this thesis as a minimal return for their efforts. I would like to thank my sister *Asli*, my mother-in-law *Serpil*, and my father-in-law *Ibrahim*, who has passed away during this work, may God be pleased with him.

TABLE OF CONTENTS

List of Tables	xiii
List of Figures	xiv
CHAPTER 1 INTRODUCTION	1
1.1 Thesis Overview	4
1.1.1 Object Tracking: Survey of the State-of-the-art	5
1.1.2 Object Tracking in Infrared Imagery Captured from Airborne Vehicles	6
1.1.3 Contour Based Object Tracking with Occlusion Handling	7
1.1.4 Spatio-Temporal Volume Sketch: A Novel Action Representation . . .	7
CHAPTER 2 OBJECT TRACKING: STATE-OF-THE-ART	9
2.1 Object Representation	12
2.2 Feature Selection for Tracking	16
2.3 Object Detection	17
2.3.1 Point Detectors	17
2.3.2 Background Subtraction	21

2.3.3	Segmentation	26
2.4	Object Tracking	31
2.4.1	Tracking by Point Correspondence	34
2.4.2	Tracking by Matching Primitive Geometric Regions	48
2.4.3	Tracking by Contour Evolution	58
2.5	Relevant Issues	65
2.5.1	Object Representation	66
2.5.2	Feature Selection	68
2.5.3	Resolving Occlusion	69
2.5.4	Limitations of Single Camera and Multiple Camera Tracking	71
2.6	Conclusions	73
CHAPTER 3 VISUAL FEATURES		74
3.1	Color	75
3.1.1	Modeling Color	77
3.2	Texture	80
3.2.1	Modeling Texture	83
3.3	Local Intensity Deviation (LID) Feature	85
3.4	Fusing Features	87

3.4.1	Cascaded Integration	87
3.4.2	Supervised Late Integration	89
3.4.3	Unsupervised Early Integration	90
3.5	Summary	92

CHAPTER 4 OBJECT TRACKING IN INFRARED IMAGERY CAPTURED FROM AIRBORNE VEHICLES 93

4.1	Tracking Model	95
4.2	Methodology	96
4.3	Object Model Update	98
4.4	Ego-Motion Compensation	100
4.5	Algorithm	105
4.6	Experiments	106
4.6.1	Object Tracking for Mobile Camera	107
4.6.2	Tracking Under High Sensor Ego-Motion	109
4.6.3	Tracking Cold Infrared Objects	110
4.6.4	Tracking Multiple Objects	110
4.7	Conclusions	111

CHAPTER 5 CONTOUR BASED OBJECT TRACKING WITH OCCLUSION HANDLING	118
5.1 Bayesian Framework For Object Tracking	120
5.2 MAP Estimation	125
5.2.1 Defining the Energy Functional	126
5.2.2 Minimizing the Tracking Functional	130
5.3 Contour Representation and Evolution	134
5.3.1 Level Set Representation	135
5.3.2 Relating Level Sets with Energy Minimization	137
5.4 Tracking During Occlusion	140
5.4.1 Occlusion Detection	141
5.4.2 Estimating the Occluded Object Contour	144
5.5 Experimental Results	146
5.5.1 Single Object Sequences	147
5.5.2 VIVID Dataset	149
5.5.3 Infrared Sequences	150
5.5.4 Occluding Object Sequences	151
5.6 Conclusions	152

CHAPTER 6 SPATIO-TEMPORAL VOLUME SKETCH: A NOVEL ACTION REPRESENTATION	164
6.1 Action Representation	169
6.1.1 Generating the Action Volume	169
6.2 The Action Sketch	177
6.2.1 Finding the Action Elements	179
6.2.2 Action Elements and Their Relations to Object Motions	182
6.3 View Invariant Action Recognition	184
6.3.1 View Invariance	185
6.3.2 Matching Actions	188
6.4 Experiments	192
6.5 Conclusions	193
CHAPTER 7 CONCLUSIONS AND FUTURE DIRECTIONS	197
7.1 Future Directions	200
CHAPTER References	202

LIST OF TABLES

2.1	Object detection categories.	18
2.2	Tracking categories.	32
2.3	Qualitative comparison of point trackers. # : number of objects, M : multiple objects, S : single object, \surd : available, \times : not available.	48
2.4	Qualitative comparison of geometric model based trackers. “Init.” denotes initialization. # : number of objects, M : multiple objects, S : single object respectively, A : affine homography, T : translational motion, S : scaling, R : rotation, P : partial occlusion, F : full occlusion.	59
2.5	Qualitative comparison of contour evolution based trackers. # : number of objects, S : single, M : multiple, P : partial, F : full, B : boundary, R : Region.	66
6.1	Fundamental surface types based on Gaussian curvature, K , and mean curvature, H	178
6.2	Recognition results for various actions. Bold face represents the correct matches and italics indicate the best match is incorrect.	194

LIST OF FIGURES

2.1	Object representations. (a) Centroid, (b) multiple points, (c) rectangular, (d) elliptical and, (e) multiple patches, (f) object contour, (g) control points on the contour, (h) object silhouette.	13
2.2	Feature points detected by applying (a) the Harris, (b) the KLT and (c) the SUSAN feature detectors.	19
2.3	Mixture of Gaussian modeling for background subtraction. (a) Image from a sequence in which a person is walking across the camera field of view. (b) The mean of the highest weighted Gaussians at each pixel position. These means represent the most temporally persistent per-pixel color and hence should represent the stationary background. (c) The means of the Gaussian with the second highest weight; these means represent colors that are observed less frequently. (d) Background subtraction result. The foreground consists of the pixels in the current frame that matched a low weighted Gaussian.	22

2.4	Eigenspace decomposition based background subtraction (space is constructed with objects in the FOV of camera), (a) an input image with objects, (b) reconstructed image after projecting input image into the Eigenspace, (c) difference image, note that the foreground objects are clearly identifiable. . .	24
2.5	Segmentation of the image shown in (a), using mean shift segmentation (b) and normalized cut (c).	27
2.6	(a) Multi-point correspondence. (b) Parametric transformation of a rectangular patch. (c,d) Two examples of contour evolution.	33
2.7	(a) All possible associations of a point (object) in frame $t - 1$ with points (objects) in frame t , (b) unique set of associations plotted with bold lines, (c) multi frame correspondences.	35
2.8	Motion constraints (a) proximity, (b) maximum velocity (r denotes radius), (c) small velocity change, (d) common motion, (e) rigidity constraints. ‘ Δ ’ denotes object position at frame ‘ $t - 2$ ’, ‘ \circ ’ denotes object position at frame $t - 1$, and finally ‘ \times ’ denotes object position at frame t	35

2.9	Row 1: A Kalman filter gives an optimal estimate of the posterior density given a Gaussian prior, and measurement densities and assuming linear dynamics. The posterior density is a Gaussian. Row 2: Particle filters can be used to propagate a general probability density function.	41
2.10	Mean shift tracking iterations. (a) Estimated object location at time $t - 1$, (b) Frame at time t with initial location estimate using the previous object position, (c), (d), (e) location update using mean shift iterations, (f) final object position at time t	51
2.11	Tracking features using the KLT tracker.	52
2.12	(a) Edge observations along the contour normals.(b) Level-set contour representation; each grid position encodes the Euclidean distance from the contour, gray level represents the value of the grid.	60
3.1	(a) RGB image. (b) r , (c) g , (d) s components of the RGS model.	76
3.2	(a) Color image. (b) Histogram (250 bins) and (c) kernel density estimate of the intensity values (gray level image).	80

3.3	Steerable pyramid representation of an image. HP refers to the high-pass filter, LP0 and LP1 refer to low-pass filter, BP0, BP1,...BPK refer to band-pass filters. The images on the right are the magnitude responses of the band-pass filters for the input image given on the left.	83
3.4	Sample texture (top image). Four boxes represent the four bandpass filter responses using Gabor wavelets in four directions. Inside each box, the first row is the magnitude of the response, second row is the phase response and last row is their mixture model computed using the EM algorithm for $C_N = 2$.	85
3.5	Local standard deviation feature: (a) input image, (b) resulting local standard deviation.	86
3.6	Cascaded integration decision flow.	88
3.7	Supervised late integration decision flow.	90
3.8	Unsupervised late integration decision flow.	91
4.1	The histogram of distances calculated using Equation 4.2, and Gaussian model fitted to the distribution.	100

4.2	Distribution of the distances computed (a) before model update, (b) after model update.	101
4.3	(a) Sample frame from one of the infrared sequences, (b) summation of four Gabor responses of the frame in (a).	102
4.4	(a) The reference frame I_k , (b) the current frame I_{k+1} , (c) the difference image obtained using (a) and (b). Note the large bright spots due to miss registration. (d) First frame registered onto the second frame, i.e., global motion is compensated, (e) the difference image obtained using (b) and (d). .	104
4.5	Tracking results for sequence rng17_01, frames 1, 17, 35, 53, 70 and 115. . . .	108
4.6	Tracking results for sequence rng14_15, frames 1, 60, 128, 195, 237 and 271. .	108
4.7	Tracking results for sequence rng19_07, frames 129, 138, 144, 159, 174 and 189.	109
4.8	The first and the second columns show results with and without global motion compensation and model update respectively. (a) Sequence rng15_20, (b) sequence rng22_08. The correct positions are marked by \times , and the detected positions are marked by \bigcirc	113

4.9	Object tracking results for cold objects: (a) sequence rng18_07, frames 113, 135, 181, 204, 229 and 259, (b) sequence rng18_03, frames 11, 39, 63, 91, 119, 154 and 170.	114
4.10	Tracking results for multiple objects in sequence rng16_07, frames 226, 241, 253, 274, 294 and 386.	115
4.11	Tracking results for multiple objects in sequence rng19_NS, frames 208, 215, 231, 253, 267 and 274.	116
4.12	Tracking results for multiple objects in sequence rng16_18, frames 1, 20, 40, 60, 79 and 99.	117
5.1	Flow chart of proposed contour tracking method.	119
5.2	Definition of object and background regions in an image.	120
5.3	(a) An object region at time t ; (b) example of a hole or (c) noise appearing at time $t + 1$	123

5.4	(a) The object region. (b) Selected band inside the object region, R_{obj}^{Γ} defined by the contour. (c) Selected band outside the object region, R_{bck}^{Γ}	124
5.5	(a) Rectangular subregion, where bold line is the object contour and m is limit of the rectangle. (b) Band around the object contour defined by the rectangular subregions of (a).	127
5.6	(a) Contour on a level set grid. (b) Distance transformation with contour superimposed.	136
5.7	Evolving contour under curvature flow. (a) 0-levels at various time slices. (b) The surface constructed by curvature based contour evolution. Note that cutting the surface at various levels will give the evolving contours given in (a). (c) Contour plot of the surface given in (b).	138

5.8	Sequence of frames with two synthetic objects (red and blue) occluding the tennis player. (a) First frame of the sequence. (b) Initial object contours super imposed on the objects. Extracted (c) player, (d) rectangular, and (e) circular objects. (f) Second frame of the sequence. (g) Tracking results using the visible features (contours are superimposed on the objects). Extracted (h) player, (i) rectangular, and (j) circular objects. Note the missing regions in (h) due to occlusion.	143
5.9	a) Contour represented on the spatial grid points, b) increasing the number of grids for the contour shown in (a), (c) consecutive level sets, ϕ'_i , from a walking person sequence with a coarser grid.	145
5.10	Synthetic occlusion sequence. (a) Tracking result using visible features (given in Figure 5.8g). (b) Tracking result using the occlusion handling method given in Equation 5.25 (contours superimposed). Note that the occludee is marked with a thin contour, whereas the occluders are marked with thick contours. Extracted (c) player, (d) rectangle and (e) ellipse. Note that the missing contour regions in Figure 5.8h are correctly estimated. (f) Selected frames from the complete sequence. The first row shows the contours superimposed and the second row shows extracted objects. Note that multiple object occlusion in the second frame of (f) is correctly handled.	154

5.11	Tracking results for the tennis player sequence acquired using a mobile camera. We recommend viewing the results in color.	155
5.12	Contour tracking results for a walking person sequence, in which the visual features both for the foreground and the background change. We recommend viewing the results in color.	156
5.13	Contour tracking results for a sequence acquired from a stationary surveillance camera.	157
5.14	Contour tracking results for walking person sequence acquired using mobile camera.	158
5.15	Contour tracking results “Passing EO” sequence for the VIVID dataset. . . .	159
5.16	Contour tracking results “Tank EO” sequence for the VIVID dataset. . . .	159
5.17	Contour tracking results “Ural EO” sequence for the VIVID dataset. . . .	160
5.18	Tracking of targets in FLIR sequences taken from an airborne platform. (a) Sequence rng14_15; (b) sequence rng16_18; (c) sequence rng23_12.	161

5.19	Contour tracking results for occluding dancers. The first row shows the frame from the sequence and the second row shows the extracted dancers. (a) Prior to occlusion, (b) occlusion starts, (c) full occlusion occurs, (d) dancer B is visible on the opposite side (partial occlusion), (e) occlusion ends. We recommend viewing these results in color.	162
5.20	Contour tracking results for two person occlusion. The first row shows frames from the sequence and the second row shows the extracted objects. (a) Prior to occlusion, (b) occlusion starts, (c) full occlusion occurs, (d) occludee is visible on the opposite side, (e) occlusion ends. We recommend viewing these results in color.	163
6.1	3D action volume of a falling person.	165
6.2	(a) A sequence of tracked object contours. (b) Clouds of voxels in the spatio-temporal space.	170
6.3	(a) Local contours from two consecutive frames used in defining the weights of central vertices. Matching of a subset of vertices between contours for (b) falling, (c) dance and (d) tennis stroke actions.	171

6.4	Visualization of spatio-temporal action volumes from two views for (a) dance, (b) tennis stroke, and (c) walking.	175
6.5	Projection of action volume. Object contours generated by fixing the (a) s and (b) t parameters in $\mathbf{B} = f(s, t)$ respectively.	176
6.6	Action volumes for (a) dance sequence with 40 frames, (b) synthetic dancer sequence with 20 frames, generated by randomly removing frames.	177
6.7	Color coded action characteristics corresponding to various surface types. Color codes are: red (peak), yellow (ridge), white (saddle ridge), blue (pit), pink (valley) and green (saddle valley).	181
6.8	Directions of motion for (a) convex wire, (b) concave wire, (c) straight wire.	183
6.9	(a) Concave contour segment denoted by three control points and (b) a normal rotated concave contour segment.	187
6.10	Associated vertices using maximum graph matching. The red vertices are from the tennis stroke action, the blue vertices are from the falling action. Some vertices have no correspondence due to 0 weight.	189

6.11	Walking action from five different view points, first row shows one frame of the action, second row shows associated contours and third row shows the corresponding action volumes. (a) 30°, (b) 60°, (c) 90°, (d) 145°, (e) 180°. (f) Matching scores, for the action in (c) with the other actions.	191
6.12	Action sketches superimposed on the action volumes, which are generated from video sequences of human actors. Numbers denote the action labels in Table 6.2. Continues in Figure 6.13.	195
6.13	Continuation of Figure 6.12.	196

CHAPTER 1

INTRODUCTION

Computer vision research started in the 1960's as an artificial intelligence problem, where the goal was to understand images by detecting objects and defining the relationship between them. Thus, in the 1970's problems such as object segmentation, and edge/line detection motivated researchers to develop methods to solve early vision problems. In 1982, David Marr [Mar82] proposed the theory of computer vision based on his background in mathematics and neuroscience. He stated that given visual information, a complete geometric reconstruction of the observed scene is required for visual perception. In his view, a vision system should compute the shape, orientation, color of every object in the scene and identify them from just one image [Mar82]. Due to the influence of this approach, during the 1980's, several researchers were involved in developing "shape from X" methods to recover the three dimensional structure of the objects, where X referred to motion [CA91], shading [YS02b], stereo [BZ92], etc.

The 1990's were the years when researchers started shifting from Marr's theory of vision to an alternate theory, known as *active vision* [BY92, AB87, SS93]. Active vision is based

on actively changing the sensor orientation and location to acquire additional information. In contrast to Marr's approach, active vision does not demand a complete representation of the environment. Rather, it defines independent visual tasks (behaviors), which have well-defined goals and operate independently to perform visual perception. An immediate result of this theory was to use simple methods to perform simple visual tasks. Although active vision boosted research in early 90's, using active sensors for visual perception was not practical.

With the advent of inexpensive high speed computers in the late 1990's, several researchers concentrated on motion-based approaches to solve computer vision problems. The goal of a motion-based approach was to interpret the scene (a sequence of frames from a video clip or a single image) based on visual cues. For instance, to perform high level vision tasks, such as action recognition [YRS02] and video retrieval [RS03], many researchers started using simpler two-dimensional representations without recovering the three-dimensional information. Compared to working on a single image, a *sequence* of frames introduces a new dimension: *time*, and a new constraint: *temporal coherence*, for the interpretation of the scene. Introduction of this additional constraint boosted the research on video processing. Video can be processed for various purposes. For instance, to obtain the story from video, one can group similar frames into shots, shots into scenes, etc., and analyze their visual contents [RS03]. Building surveillance systems was another motivation for video processing.

Due to the challenges involved and the scope of application, surveillance has quickly become a very popular research topic in the past few years [HHD00, JS02, ED01]. The basic components of a surveillance system are:

1. *Object detection* which finds region of interests (moving or stationary) in an image.
2. *Object tracking* which performs tracking and generates trajectories.
3. *Object classification* which classifies tracked objects as car, suitcase, human, truck, tank, etc.
4. *Activity recognition* which analyzes the activities and behaviors performed by individual objects or group of objects.

The tracking component is usually a minimal requirement for a surveillance system, and one of the most important bottlenecks. If the object tracker fails, then the rest of the system becomes unreliable. Recognizing objects and their actions is also very important for a visual surveillance system. In this thesis, we address problems related to tracking objects in video acquired using mobile cameras, propose novel tracking approaches to overcome these problems and introduce a novel representation for actions performed by actors for behavior recognition.

1.1 Thesis Overview

Despite the tremendous amount of research on object tracking, there is no comprehensive survey of visual tracking. Following this observation, we provide a comprehensive survey and categorization of the state-of-the-art tracking methods proposed in the recent years (see Chapter 2). Chapter 3 discusses visual features used in tracking methods, such as the color and the texture features, and details on possible modeling approaches used by various researchers. In Chapters 4 and 5, we propose two novel approaches for tracking objects. For both of these tracking methods, our emphasis is on high inter-frame motion and moving camera, which remain a challenge for most object trackers. The first method uses the primitive geometric regions to represent the object and the second method uses the contour representation. Both of these methods have advantages and disadvantages. For instance, our first method can perform tracking in real time with very high accuracy, however it only tracks a circular (or ellipsoidal) window on the target (not necessarily centered on the object centroid). On the other hand, the second method tracks the full object, resolves object occlusions, and therefore can be used in high-level processes, such as action recognition, object identification and object recognition. Finally, in Chapter 6, we propose a novel and compact representation for actions performed by tracked objects. The proposed action representation uses the contours tracked by the contour tracking method discussed in Chapter 5 to generate a continuous volume. Action features are extracted by analyzing the differential geometric properties of the underlying volume. In Chapter 7.1, we conclude our thesis and

provide future research directions. Below we provide an overview of the proposed trackers and action representation.

1.1.1 Object Tracking: Survey of the State-of-the-art

Object tracking, in general, is a challenging problem. Difficulties in tracking objects can arise due to abrupt object motion, changing appearance patterns of both the object and the scene, non-rigid object structures, object to object and object to scene occlusions, and camera motion. Tracking is usually performed in the context of higher level applications that require the location or shape of the object. The assumptions made to constrain the tracking problem are employed in the context of that application. In this chapter, we categorize the tracking methods on the basis of the object and motion representations used. We provide detailed descriptions of representative methods in each category and examine their pros and cons. Moreover, we discuss the topics relevant to tracking in general, such as the use of appropriate image features, selection of motion models, and detection of objects.

1.1.2 Object Tracking in Infrared Imagery Captured from Airborne Vehicles

Small and quickly moving objects in video acquired by airborne mobile platforms are usually hard to track. In addition, rapid motion of the sensor creates irregular object motion, which violates constant speed and acceleration constraints. We propose an object tracker to overcome these problems in closing sequences of forward looking infrared-imagery (FLIR), which is motivated by the mean shift tracker proposed in [CRM00]. The main contributions of our object tracker are:

- We use more features to model the object correctly. In particular, the density estimations of local standard deviation features are fused with the intensity models. Due to the target's low contrast with the background, extra features are necessary for FLIR imagery.
- We use additional tools, such as a global motion estimator to relax the small object motion constraint imposed by the object trackers in this category (see Section 2.4.2). The tracking module automatically decides whether motion compensation is necessary.
- The object feature models are automatically updated to adapt to rapid changes in the visual features due to noise introduced from the IR sensor.

1.1.3 Contour Based Object Tracking with Occlusion Handling

We propose a contour tracking method for video acquired using a mobile camera, which tracks the complete objects. The proposed method can track multiple objects, adapt to changing visual features, and handle partial and full object occlusions. Tracking is achieved by evolving the contour from frame to frame by minimizing an energy functional, which is formulated using variational calculus. We minimize the energy in the gradient descent direction evaluated in the contour vicinity defined by a band. Our approach has two major components related to the visual features and the object shape. Visual features (color, texture) are modeled by semi-parametric models, and their relative weights are computed using independent opinion polling. The shape prior is used to recover the missing object regions during occlusion. Each shape prior consists of a shape level set, where each cell in the level set grid has an associated Gaussian. We demonstrate the performance of our method on real sequences with and without object occlusions.

1.1.4 Spatio-Temporal Volume Sketch: A Novel Action Representation

Once object contours are available from the proposed contour tracking approach, action recognition can be performed. Actions can be represented by changes in speed, direction and shape of the performing object. Changes in these quantities stem from the motion of

the object in the three-dimensional space. We propose modeling important events during an action based on the motion of the object in space and time using “action volumes”. Action volumes can be generated by tracking the points on a three-dimensional object performing an action, which results in tracks in four dimensions (x, y, z, t) . Under the orthographic projection, space-time trajectories becomes spatio-temporal trajectories in (x, y, t) . A set of spatio-temporal trajectories construct a solid, which we call a “spatio-temporal action volume”. In contrast to tracking one or a few points on the object, we generate the action volume by tracking the complete object contour. Once tracking is performed, correspondences between contours in different frames are determined using a dynamic programming approach. Important events during an action (action elements) are computed by analyzing the properties of the local surfaces of the action volume, such as peaks, pits and ridges. Action elements computed using surface analysis are invariant to the position of the camera. We relate these view invariant surface properties to various types of motions. Finally, using these features, we perform view invariant action recognition.

CHAPTER 2

OBJECT TRACKING: STATE-OF-THE-ART

Object tracking is an important task within the area of computer vision. The proliferation of high powered computers and the increasing need for automated surveillance systems have generated a great deal of interest in object tracking algorithms. Some of the tasks that use object tracking are:

- Motion-based recognition: human identification based on gait, automatic object detection, etc.,
- Automated surveillance: monitoring a scene to detect suspicious activities or unlikely events,
- Video indexing: automatic annotation and retrieval of videos for multimedia databases,
- Human-computer interaction: gesture recognition, eye gaze tracking for data input to computers,
- Traffic monitoring: real time gathering of traffic statistics to direct traffic flow, and

- Vehicle navigation: navigating vehicles with path planning and obstacle avoidance capabilities.

Tracking can be defined as the problem of estimating the trajectory of an object as the object moves around a scene. Simply stated, we want to know where the object is in the image at each instant in time. Tracking objects can be a complex problem due to:

- loss of information caused by projection of the 3D world on a 2D image,
- noise in images,
- complex object motion,
- partial and full object occlusions,
- complex object shapes
- scene illumination changes and
- real time processing requirements.

One can simplify the tracking problem by imposing constraints on the motion and appearance of objects. For example, almost all tracking algorithms assume the object motion to be smooth. One can further constrain the object motion to be of constant velocity or constant acceleration based on a priori information. Prior knowledge about the number and the size of the objects can also be used to simplify the problem. Some algorithms assume prior

knowledge about the object appearance or shape, e.g., tracking skin colored objects, or tracking elliptical objects.

Numerous approaches for object tracking have been proposed. These primarily differ from each other based on the way they tackle the following questions: Which object representation is suitable for tracking? Which image features should be used? How should the motion, the appearance and the shape of the object be modeled? The answers to these questions depend on the context/environment in which the tracking is being performed, and the end use for which the tracking information is being sought. A large number of tracking methods have been proposed that attempt to answer these questions for a variety of scenarios. The goal of this chapter is to group tracking methods into appropriate categories and provide comprehensive descriptions of representative methods in each category. We aspire to provide the readers that require a tracker for a certain application with the ability to select the most suitable tracking algorithm for their particular needs. Moreover, we aim to identify new trends and ideas in the tracking community and hope to provide insight for the development of new tracking methods. Finally, we want to point out that the focus of this chapter is on methodologies for tracking objects in general, and not on trackers tailored for specific objects, e.g., people trackers.

Our discussion follows a bottom-up approach in describing the issues that need to be tackled when one sets out on a task of building an object tracker. The first issue is coming up with a suitable representation of the object. In Section 2.1, we will describe some common object shape representations, e.g., points, primitive geometric shapes and object

contours, and appearance representations. The next issue is the selection of image features used as an input for the tracker. In Section 2.2, we discuss various image features, like color, motion and edges, which are commonly used in object tracking. Almost all tracking algorithms require detection of the objects, either in the first frame or in every frame depending on the tracking method used. Section 2.3 summarizes the general strategies for detecting the objects in a scene. The suitability of a particular tracking algorithm depends on object appearance, object shape, number of objects, the object and camera motion, and the illumination conditions. In Section 2.4, we categorize and describe the existing tracking methods and also explain their strengths and weaknesses in a summary section at the end of each category. Section 2.5 discusses important issues relevant to object tracking.

2.1 Object Representation

Objects can be represented by their shapes and appearances. In this section, we will first describe object shape representations commonly employed for tracking and then address the joint shape and appearance representations.

- **Point:** The object is represented by a point, e.g., the centroid (Figure 2.1a). In general, the point representation is suitable for tracking objects that occupy very small regions in an image.(see Section 2.4.1).

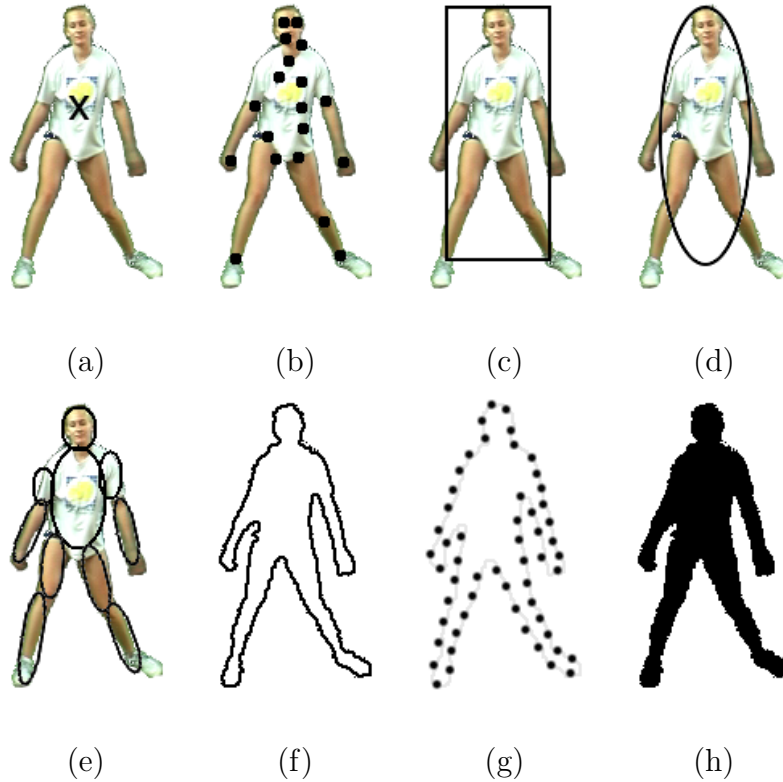


Figure 2.1: Object representations. (a) Centroid, (b) multiple points, (c) rectangular, (d) elliptical and, (e) multiple patches, (f) object contour, (g) control points on the contour, (h) object silhouette.

- Primitive geometric shapes:** The object shape is represented by a rectangle, ellipse, etc. (Figure 2.1c, d). Object motion, for such representations, is usually modeled by translation, affine or projective transformation (see Section 2.4.2 for details). Though primitive geometric shapes are more suitable for representing simple rigid objects, they are also used for tracking a variety of non-rigid objects.

- **Object contour:** The contour representation defines the boundary of the object and is suitable for representing non-rigid objects (Figure 2.1f, g). The region inside the contour is the silhouette of the object (see Figure 2.1h). Contour based representations are suitable for tracking complex non-rigid shapes.
- **Articulated shape models:** Articulated objects are composed of object parts that are held together with joints. For example, the human body is an articulated object with torso, legs, hands, head and feet connected by joints. The relationships between the parts are governed by the model parameters, e.g., joint angle. To represent the constituent parts of the articulated model, one can use lines, ellipses (Figure 2.1e) or cylinders. Articulated object tracking is beyond the scope of this thesis and we refer the interested reader to the surveys [AC99, Gav99, MG01] on this topic.

There are a number of ways to represent the appearance features of objects. Note that shape representations can also be combined with the appearance representations for tracking. Some common appearance representations in the context of tracking algorithms are given as follows:

- **Probability densities of object appearance:** The probability density estimates of the object appearance can either be parametric, e.g., a Gaussian [ZY96] or a Gaussian mixture model [PD02], or non-parametric, e.g., Parzen windows [EDH02] or histograms [CRM03]. The probability densities of object appearance features, e.g., color, can be

computed from the image regions specified by the shape models, e.g., interior region of an ellipse or a contour.

- **Templates:** Templates are formed using simple geometric shapes or silhouettes [FT97]. An obvious advantage of a template is that it carries both spatial and appearance information. The limitation of template based models is that they only encode appearance of the objects from one view. Thus, they are only suitable for tracking objects whose pose does not vary considerably.
- **Multi-view appearance models:** These models encode different views of an object. For example, we can represent all the views of an object by a subspace decomposition, e.g., Eigenspace decomposition [BJ98]. Another approach to model the appearance is to train classifiers, e.g., support vector machines [Avi01], to represent the different views of the object. One limitation of this approach is that the appearance information of all views of the tracked objects is required ahead of time.

Object representations are usually chosen according to the tracking application. For tracking objects which appear very small in an image, point representation is usually appropriate. For instance, Veenman et al. [VRB01] use point representation to track the seeds in a moving dish sequence, and Shafique and Shah [SS03] use point representation to track birds. If the objects have shapes like a rectangle or an ellipse, primitive geometric shape representations are more appropriate. Comaniciu et al. [CRM03] use an elliptical shape representation and employ a color histogram computed from the elliptical region for appearance

modeling. In [BJ98], Black and Jepson use the Eigenspace based appearance representation, which is generated from the rectangular object regions. For objects with complex shapes, e.g., humans, a contour or a silhouette based representation is appropriate. Haritaoglu et al. [HHD00] use object templates in the form of object silhouettes for object tracking in a surveillance application. Generally, there is a strong relationship between the object representation and the tracking algorithms. We will have a further discussion (in Section 2.5.1) on selection of the right representation for tracking after we discuss the tracking methodologies.

2.2 Feature Selection for Tracking

Tracking methods requires a set of unique features to represent each tracked object. Selecting the right features is closely related to the object representation. For example, color is used as a feature for histogram based appearance representations while for contour based representation, edges are usually used as features. Common features used in tracking methods are:

- color,
- edges,
- optical flow,
- texture.

Color is one of the most widely used features for tracking. Comaniciu et al. [CRM03] use color as a feature for the histogram representation of the tracked object. Cremers et al. [CS03] use optical flow as a feature for contour based tracking. Jepson et al. [JFE03] use steerable filter responses as features. For a detailed discussion on these features, please see Chapter 3.

2.3 Object Detection

Every tracking method requires an object detection mechanism either in every frame or when the object first appears in the video. Some common object detection methods are given in Table 2.1. Although object detection itself requires a survey of its own, here we outline some of the popular methods for the sake of completeness.

2.3.1 Point Detectors

Point detectors, which find interest points in an image, have been successfully used in the context of motion, stereo and tracking for a long time. An interest point, e.g., a corner in an image, ideally should be invariant to the changes in illumination and camera view. A property of interest points is that there is expressive texture in the locality of the point. Commonly

Table 2.1: Object detection categories.

Categories	Representative work
Point detectors	Moravec’s detector [Mor79], Harris detector [HS88], Affine Invariant Point Detector [MS02]
Segmentation	Mean shift [CM99] Graph-cut [SM00], Active contours [CKS95]
Background Modeling	Mixture of Gaussians[SG00], Eigenbackground[ORP00], Wall flower [TJM99], Dynamic texture background [MMP03]

used interest point detectors include Moravec’s interest operator [Mor79], Harris interest point detector [HS88], KLT detector [ST94], and SUSAN detector [SB97]. Moravec’s interest operator computes the variation of the image intensities in a 4x4 patch in the horizontal, vertical, diagonal and anti-diagonal directions, and selects the minimum of the four variations as representative value for the window. A point is declared interesting if the intensity variation is a local maximum in a 12x12 patch.

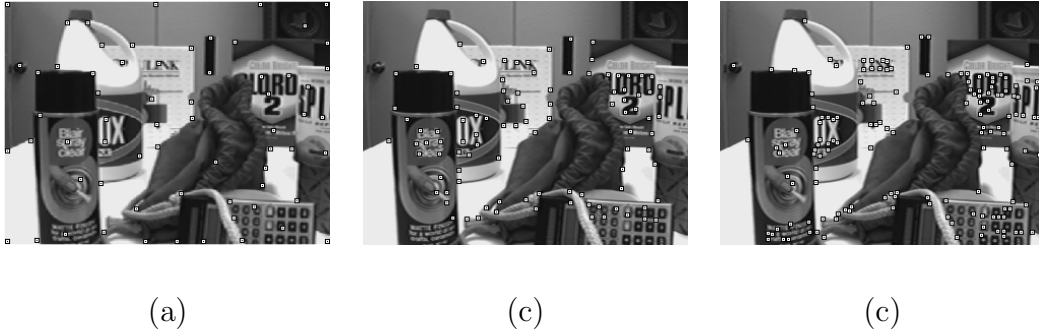


Figure 2.2: Feature points detected by applying (a) the Harris, (b) the KLT and (c) the SUSAN feature detectors.

The Harris detector computes the first order image derivatives, (I_x, I_y) , in x and y directions to highlight directional intensity variations, then a second moment matrix, which encodes this variation, is evaluated for each pixel: $M = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$. An interest point is identified from the determinant and the trace of M which measure the variation in a local neighborhood: $R = \det(M) - k \cdot \text{tr}(M)^2$ where k is constant. The interest points are marked by thresholding R after applying non-maxima suppression (see Figure 2.2a for results). The source code for the Harris detector is available at [har]. Similar to the Harris detector, in the KLT detector, a moment matrix M is computed but in a 6×6 patch. Interest point confidence, R , is considered from the minimum eigenvalue of M , λ_{min} . Interest point candidates are selected by thresholding R . Among the candidate points, KLT eliminates the candidates that are spatially close to each other (Figure 2.2b). Implementation of the KLT detector is available at [klt].

Quantitatively both Harris and KLT emphasize the variations using very similar measures. For instance, R in Harris is related to the characteristic polynomial used for finding eigenvalues of M : $\lambda^2 + \det(M) - \lambda \cdot \text{tr}(M) = 0$. Note that KLT computes the eigenvalues directly. In practice, both of these measures find the same interest points, however it is the neighborhood criterion that differentiates the KLT detector over the Harris detector.

In contrast to the aforementioned methods that compute intensity variation, SUSAN detector evaluates intensity correlation by counting the number of pixels, N , that are similar to each image pixel in a 7x7 patch. The interest points are subsequently extracted by thresholding N at each pixel followed by non-maxima suppression. In Figure 2.2c, we show the features detected by the SUSAN detector. The implementation of the SUSAN detector is available at [sus].

The Harris point detector is invariant to rotation but not to affine transformations. Mikolajczyk and Schmid [MS02] propose an affine invariant interest point detector. In the first step of the algorithm, a set of interest points are selected using the Harris detector. Next, an iterative procedure is applied that selects the best scale for computing M and then (affine) transforms the region around the point such that both the eigenvalues of M are equal. Note that after this transformation, the intensity patterns around the interest point location are isotropic. Any two isotropic regions are related by a pure rotation, therefore any rotation invariant detector will be able to detect such transformed interest points. The source code for this interest point detector is available at [Aff].

In summary, all point detectors aim to measure the variation of intensity over small image regions. Autocorrelation or the second moment matrix of intensity gradient are usually used to estimate the variation. However, these measures are not invariant to changes image scale or affine and projective transformations. In order to overcome these problems, Mikolajczyk and Schmid [MS02] use the moment matrix to normalize the region in an affine invariant way and employ an affine adapted Harris detector to determine the location of interest points. For a comparative evaluation of interest point detectors, we refer the reader to the survey by Mikolajczyk and Schmid [MS03].

2.3.2 Background Subtraction

Object detection can be achieved by building a representation of the scene called the background model and then finding deviations from the model for each incoming frame. Any significant change in an image region from the background model signifies a moving object. The pixels constituting the regions undergoing change are marked for further processing. Usually a connected component algorithms is applied to obtain connected regions corresponding to the objects. The term “background subtraction” is used to denote this process.

Frame differencing of temporally adjacent frames has been well studied since the late 70s [JN79]. However, background subtraction became popular recently after the work of Wren et al. [WAP97]. In order to learn gradual changes over-time, Wren et al. proposed to model

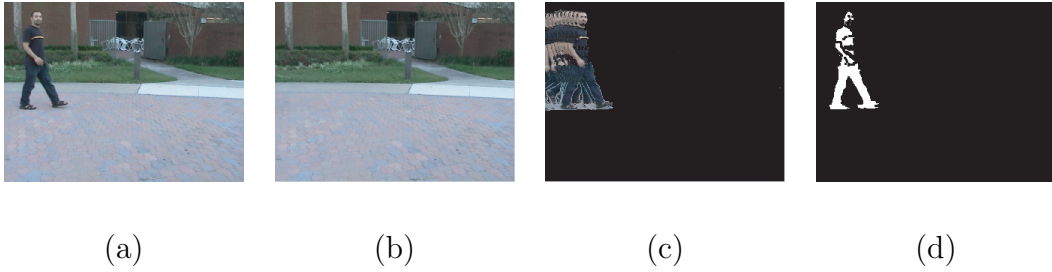


Figure 2.3: Mixture of Gaussian modeling for background subtraction. (a) Image from a sequence in which a person is walking across the camera field of view. (b) The mean of the highest weighted Gaussians at each pixel position. These means represent the most temporally persistent per-pixel color and hence should represent the stationary background. (c) The means of the Gaussian with the second highest weight; these means represent colors that are observed less frequently. (d) Background subtraction result. The foreground consists of the pixels in the current frame that matched a low weighted Gaussian.

the color of each pixel, $I(x, y)$, of a stationary background with a single 3D (Y, U and V color space) Gaussian, $I(x, y) \sim N(\mu(x, y), \Sigma(x, y))$. The model parameters, the mean $\mu(x, y)$ and the covariance $\Sigma(x, y)$ are learned from the color observations in several consecutive frames. Once the background model is derived, for every pixel (x, y) in the input frame, the likelihood of its color coming from $N(\mu(x, y), \Sigma(x, y))$ is computed, and the pixels that deviate from the background model are labelled as the foreground pixels. However, a single Gaussian is not a good model [GBC00] for outdoor scenes, since multiple colors may be observed at a certain location due to repetitive object motion, shadows or reflectance. A substantial improvement in background modeling is achieved by using multi-modal statistical models to describe the

per-pixel background color. For instance, Stauffer and Grimson [SG00] use a mixture of Gaussians to model the pixel color. In this method, a pixel in the current frame is checked against the background model by comparing it with every Gaussian in the model until a matching Gaussian is found. If a match is found the mean and variance of the matched Gaussian are updated, otherwise a new Gaussian with the mean equal to the current pixel color and some initial variance is introduced into the mixture. Each pixel is classified based on whether the matched distribution represents the background process. Moving regions, which are detected using the latter approach, along with the background models are shown in Figure 2.3.

An alternate approach for background subtraction is to represent the intensity variations of a pixel in an image sequence as discrete states corresponding to the events in the environment. For instance, for tracking cars on a highway, image pixels can be in the background state, the foreground (car) state or the shadow state. Rittscher et al. [RKJ00] use Hidden Markov Model (HMM) to classify small blocks of an image as belonging to one of these three states. In the context of detecting light on and off events in a room, Stenger et al. [SRP01] use HMM's for the background subtraction. The advantage of using HMM's is that certain events that are hard to model correctly using unsupervised background modeling approaches, can be learned using training samples.

Instead of modeling the variation of individual pixels Oliver et al. [ORP00] propose a holistic approach using the Eigenspace decomposition. For k input frames, $I^i : i = 1 \dots k$, of size $n \times m$, a background matrix \mathbf{B} of size $k \times l$ is formed by cascading m rows in each

frame one after the other, where $l = (n \times m)$, and eigenvalue decomposition is applied to the covariance of \mathbf{B} , $\mathbf{C} = \mathbf{B}^T \mathbf{B}$. The background is then represented by the most descriptive η eigenvectors, \mathbf{u}_i , where $i < \eta < k$, that encompass all possible illuminations in the field of view (FOV). Thus, this approach is less sensitive to illumination. The foreground objects are detected by projecting the current image to the Eigenspace and finding the difference between the reconstructed and actual images. We show detected object regions using the Eigenspace approach in Figure 2.4. One limitation of this approach is that it requires relatively static background.

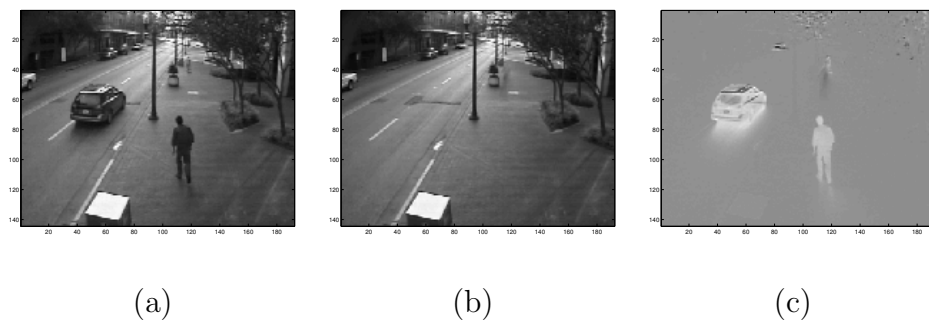


Figure 2.4: Eigenspace decomposition based background subtraction (space is constructed with objects in the FOV of camera), (a) an input image with objects, (b) reconstructed image after projecting input image into the Eigenspace, (c) difference image, note that the foreground objects are clearly identifiable.

The limitation of a static background model was recently addressed by Monnet et al. [MMP03], and Zhong and Sclaroff [ZS03]. Both of these methods are able to deal with time varying background, for example, waves on the water, moving clouds and escalators. These

methods model the image regions as autoregressive moving average (ARMA) processes. An ARMA process is a time series model that is made up of sums of autoregressive and moving-average components. Note that an autoregressive process is a process that can be described as a weighted sum of its previous values and a white noise error. These approaches use the ARMA processes to learn and predict the motion patterns in a scene.

In summary, most state-of-the-art tracking methods for fixed cameras, e.g., [HHD00, CLF01] use background subtraction methods to detect regions of interest. This is because the recent subtraction methods are able to model model the changing illumination, noise, and the periodic motion of the background regions and therefore can accurately detect objects in a variety of circumstances. Moreover, these methods are computationally efficient. In practice, background subtraction provides incomplete object regions in many instances, ie., the objects may be split into several regions or there may be holes inside the object since there are no guarantees that the object features will be different from the background features. The most important limitation of background subtraction is the requirement for stationary cameras. Camera motion usually distorts the background models. An extension of these methods to mobile camera can be obtained by regenerating background models for a small temporal windows, e.g., three frames, from scratch [KCL98] or by compensating sensor motion, e.g., creating background mosaics, [RB96, IA98a]. However, both of these solutions require that the motion in successive frames is small and can be modeled using the parametric motion models.

2.3.3 Segmentation

The aim of an image segmentation algorithm is to partition the image into perceptually similar regions. Every segmentation algorithm addresses two problems, the criteria for a good partition and the method for efficient partitioning [SM00]. In this section, we will discuss recent segmentation techniques that are related to object tracking.

2.3.3.1 Mean Shift Clustering

For image segmentation, Comaniciu and Meer [CM02] propose the mean shift approach to find clusters in the joint spatial+color space, $[l, u, v, x, y]$, where $[l, u, v]$ represents the color and $[x, y]$ represents the spatial location. Given an image, the algorithm is initialized with a large number of hypothesized cluster centers randomly chosen from the data. Then, each cluster center is moved to the mean of the data lying inside the multi-dimensional ellipsoid centered on the cluster center. The vector defined by the old and the new cluster centers is called the *mean shift vector*. The mean shift vector is computed iteratively until the cluster centers do not change their positions. Note that during the mean shift iterations, some clusters may get merged. In Figure 2.5b, we show the segmentation using the mean shift approach generated using the source code available at [meaa].

Mean shift clustering is scalable to various other applications such as edge detection, image regularization [CM02] and tracking [CRM03]. Mean shift based segmentation requires

fine tuning of various parameters to obtain better segmentation, for instance, the bandwidths of the color and the spatial kernels, and the minimum region threshold. Another limitation of the mean shift segmentation is its high computational cost. However, in the context of tracking mean shift has a real-time performance (see Section 2.4.2 for details on tracking).

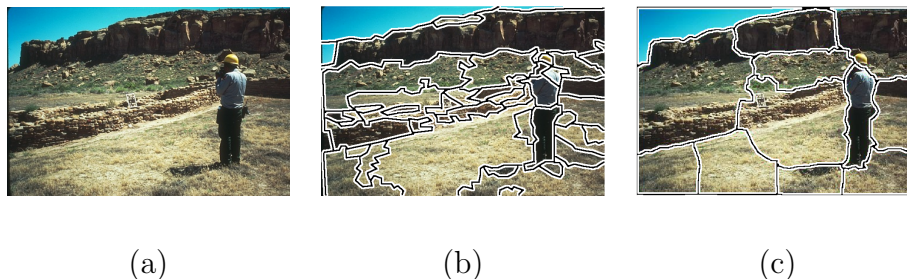


Figure 2.5: Segmentation of the image shown in (a), using mean shift segmentation (b) and normalized cut (c).

2.3.3.2 Image Segmentation Using Graph-Cut

Image segmentation can also be formulated as a graph partitioning problem, where the vertices (pixels), $\mathbf{V} = \{u, v, \dots\}$, of a graph (image), \mathbf{G} , are partitioned into N disjoint subgraphs (regions), A_i , $\bigcup_{i=1}^N A_i = \mathbf{V}$, $A_i \cap A_j = \emptyset$, $i \neq j$, by pruning the weighted edges of the graph. The total weight of the pruned edges between two subgraphs is called the “cut”. The weight is typically computed by the color, the brightness or the texture similarity between the nodes. Wu and Leahy [WL93] use the minimum cut criterion, where the goal

is to find the partitions that minimize the cut. They define the weights based on color similarity. Wu and Leahy's method is biased toward over-segmenting the image, due to the increase in cost of a cut with the number of edges going across the two partitioned segments.

Shi and Malik [SM00] propose the *normalized cut* to overcome the over-segmentation problem. In their approach, the cut not only depends on the sum of edge weights in the cut, but also on the total connection weight of nodes in each partition to all nodes of the graph. For image based segmentation, the weights between the nodes are defined by the product of the color similarity and the spatial proximity. Once the weights between each pair of nodes are computed, a weight matrix \mathbf{W} and a diagonal matrix \mathbf{D} , where $\mathbf{D}_{i,i} = \sum_{j=1}^N \mathbf{W}_{i,j}$ are constructed. The segmentation is performed first by computing the eigenvectors and the eigenvalues of the generalized Eigensystem $(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$, then the second smallest eigenvector is used to divide the image into two segments. For each new segment, this process is recursively performed until a threshold is reached. In Figure 2.5c, we show the segmentation result of the normalized cut approach.

In normalized cut based segmentation, the solution to the generalized Eigensystem for large images can be expensive in terms of processing and memory requirements. However, this method requires fewer manually selected parameters, compared to mean shift segmentation. Normalized cuts have been used in context of tracking object contours [XA02].

2.3.3.3 Active Contours

In an active contour framework, segmentation is achieved by evolving a closed contour to the object's boundary, such that the contour tightly encloses the object region. Evolution is governed by energy functionals which define the fitness of the contour to the hypothesized object region. Contour energy functionals have the following form:

$$E(\Gamma) = \int_0^1 E_{int}(\mathbf{v}) + E_{im}(\mathbf{v}) + E_{ext}(\mathbf{v}) ds,$$

where s is the arc length of the contour Γ , E_{int} includes regularization constraints, E_{im} includes appearance based energy and E_{ext} specifies additional constraints. E_{int} usually includes a curvature term, first order ($\nabla \mathbf{v}$) or second order ($\nabla^2 \mathbf{v}$) continuity terms to find the shortest contour. Image based energy, E_{im} , is commonly computed from the image gradient which is evaluated around the contour [KWT88, CKS95], or the color [ZY96, YLS04, Ron94] and texture [PD02] information which is evaluated inside and outside the object region.

Different variations of the energy functional also exist. In [CKS95], Caselles et al. use $E(\Gamma) = \int_0^1 E_{int}(\mathbf{v})E_{im}(\mathbf{v})ds$, where $E_{im} = g(|\nabla I|)$, g is the sigmoid function, and E_{int} includes first order energy terms. The authors use the level set contour representation. Paragios and Deriche [PD02] use $E(\Gamma) = \int_0^1 \lambda E_{boundary}(\mathbf{v}) + (1 - \lambda)E_{region}(\mathbf{v})ds$, where both E_{region} and $E_{boundary}$ are simply terms of E_{image} . The authors model the appearance in E_{region} by a mixture of Gaussians.

An important issue in contour based methods is the contour initialization. A common approach is to place the contour outside the object region and shrink until the object boundary

is encountered [KWT88, CKS95]. This constraint is relaxed in region based methods, such that the contour can be initialized either inside or outside the object, so that the contour can either expand or shrink respectively, to fit the object boundary. However, these approaches usually require prior object or background knowledge [PD02]. Using multiple frames or a reference frame, initialization can be performed without building region priors. For instance in [PD00], the authors used background subtraction to initialize the contour.

Besides the selection of the energy functional and the initialization, another important issue is to select the right contour representation. The contour, Γ , is represented either explicitly (control points, \mathbf{v}) or implicitly (level sets, ϕ). In the explicit representation, the relation between the control points are defined by the spline equations. In the level set representation, the contour is represented on a spatial grid which encodes the signed distances of the grids from the contour with opposite signs for the object and the background regions. The contour is evolved by modifying the grid values. Source code for a generic level set, which can be used for various applications by specifying the contour evolution speed, e.g., segmentation, tracking or heat flow, is available at [lev]. Both of these representations have their advantages and disadvantages. For instance, the most important advantage of the implicit representation over the explicit representation is its flexibility to allow topology changes (split and merge). However, due to representing the contour on a grid, contour evolution using implicit representation is computationally more expensive than the explicit representation.

2.4 Object Tracking

Tracking an object aims at locating the object and finding the region it encompasses in the image at every time instant. These two operations can be performed separately or jointly. For instance, first the regions of interest can be found by a detection algorithm and then they can be corresponded with the objects previously observed. Alternatively, the regions corresponding to the objects can be estimated iteratively, given the previous location and some similarity criteria. In either case, it is necessary to represent the region that the object encompasses using one of the shape models described in Sec. 2.1. The modeling of object motion directly depends on its representation. For example, if an object is represented as a point then only a translational model completely defines its motion. If a geometric shape representation is used then parametric motion models like affine or projective transformations can be used. These representations can approximate the motion of rigid objects in the scene. For non-rigid objects, contours are the most descriptive representation and both parametric and non-parametric models can be used to specify their motion.

In view of the aforementioned discussion, we have broadly classified tracking approaches into three categories (see Table 2.2):

- *Tracking by point correspondence:* Objects detected in consecutive frames are associated based on the previous object state, which can include motion and shape charac-

Table 2.2: Tracking categories.

Categories	Representative work
<p><i>Tracking by Point Correspondence</i></p> <ul style="list-style-type: none"> • Deterministic methods • Statistical methods 	<p>MGE tracker [SS90], GOA tracker [VRB01] Kalman filter [BC86], JPDAF [BF88], PMHT [SL94]</p>
<p><i>Tracking by matching primitive geometric regions</i></p> <ul style="list-style-type: none"> • Template and density based appearance models • Multi-view appearance models 	<p>Mean shift [CRM03], KLT [ST94], Layering [TSK02] Eigentracking [BJ98], SVM tracker [Avi01]</p>
<p><i>Tracking by contour evolution</i></p> <ul style="list-style-type: none"> • State space models • Direct minimization 	<p>Condensation [IB98], JPDAF+HMM [CRH01] Variational methods [BSR00] Heuristic methods [Ron94]</p>

teristics of the object. This approach requires an external mechanism to detect the objects in every frame. An example of point correspondence is shown in Figure 2.6a.

- *Tracking by matching primitive geometric regions:* These methods usually use a combination of shape and appearance to model the object. For example an object can be represented by a rectangular template, or by an elliptical shape and an associated histogram. Objects are tracked by estimating the parametric transformation of the shape model in consecutive frames (Figure 2.6b). Object detection is required only in the first frame for these methods.
- *Tracking by contour evolution:* Tracking is performed by estimating the object contour, given an initial contour from the previous frame. This essentially can be considered as segmentation of each frame using priors obtained from previous images. Two examples of contour evolution are shown in Figure 2.6c,d.

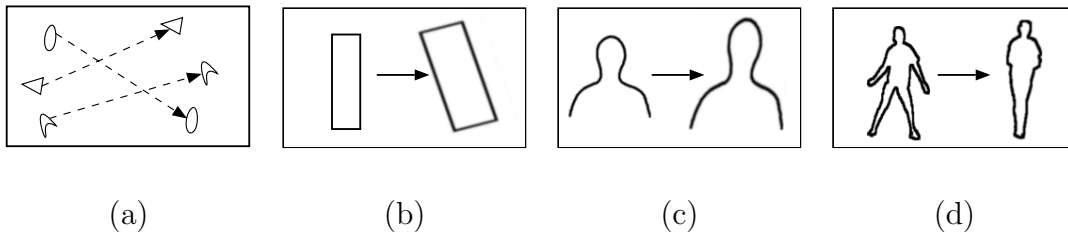


Figure 2.6: (a) Multi-point correspondence. (b) Parametric transformation of a rectangular patch. (c,d) Two examples of contour evolution.

2.4.1 Tracking by Point Correspondence

Tracking can be formulated as correspondence of detected objects represented by points across frames. Point correspondence is a complicated problem, specially in the presence of occlusions, mis-detections, entries and exits of objects. Overall, the object correspondence approaches can be divided into two broad categories, namely deterministic methods and statistical methods. The deterministic methods use “qualitative motion heuristics” [VRB01] to constrain the correspondence problem. The correspondence solution is found through a combinatorial optimization scheme. On the other hand, probabilistic methods explicitly take the object measurement and model uncertainties into account to establish correspondence. The object state consists of object kinematics such as position, velocity and acceleration. A *Maximum a Posteriori* (MAP) estimate of the object state is calculated at each time instant. Below, we describe the tracking approaches belonging to these two sub-categories in detail.

2.4.1.1 Deterministic Methods for Correspondence

The deterministic methods for correspondence define a cost of associating each object in frame $t - 1$ to a single object in frame t using a set of motion constraints. Minimization of the correspondence cost is formulated as a combinatorial optimization problem. A solution, which consists of one-to-one correspondences (Figure 2.7b) among all possible associations

(Figure 2.7a), can be obtained by optimal assignment methods, e.g., the Hungarian algorithm, [Kuh55] or greedy search methods.

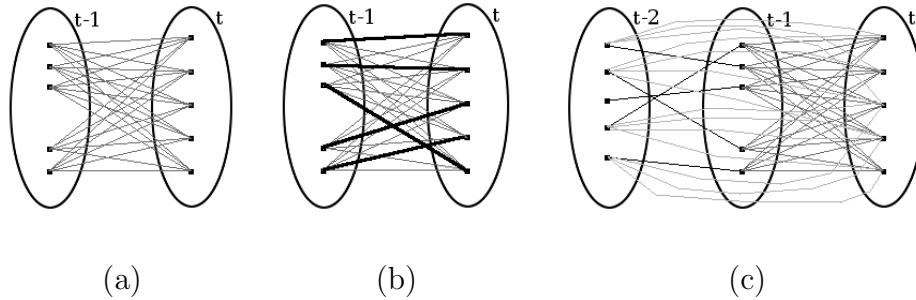


Figure 2.7: (a) All possible associations of a point (object) in frame $t - 1$ with points (objects) in frame t , (b) unique set of associations plotted with bold lines, (c) multi frame correspondences.

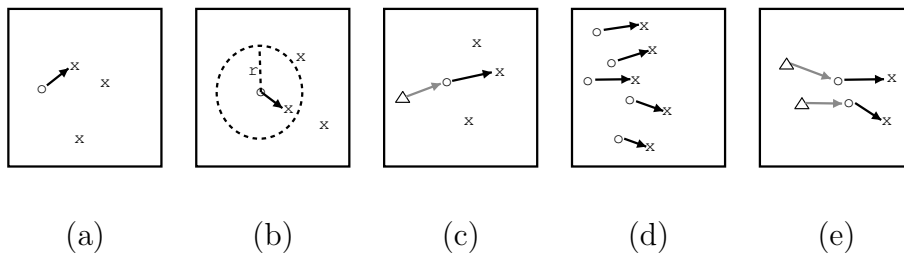


Figure 2.8: Motion constraints (a) proximity, (b) maximum velocity (r denotes radius), (c) small velocity change, (d) common motion, (e) rigidity constraints. ' Δ ' denotes object position at frame ' $t - 2$ ', ' \circ ' denotes object position at frame $t - 1$, and finally ' \times ' denotes object position at frame t .

The correspondence cost is usually defined by using a combination of the following constraints:

- *Proximity* assumes the location of the object would not change notably (see Figure 2.8a).
- *Maximum velocity* defines an upper bound on the object velocity, and minimizes the number of possible correspondences to the circular neighborhood around the object (see Figure 2.8b).
- *Small velocity change* (smooth motion) assumes the direction and speed of the object does not change drastically (see Figure 2.8c).
- *Common motion* constrains the velocity of objects in a small neighborhood to be similar (see Figure 2.8d). This constraint is suitable for objects represented by multiple points.
- *Rigidity* assumes that objects in the 3D world are rigid, therefore the distance between any two points on the actual object will remain unchanged (see Figure 2.8e).
- *Proximal uniformity* is a combination of the proximity and the small velocity change constraints.

In particular, Sethi and Jain [SJ87] solve the correspondence by a greedy approach based on the proximity and rigidity constraints. Their algorithm considers two consecutive frames, and is initialized by the nearest neighbor criterion. The correspondences are exchanged iteratively to minimize the cost. A modified version of the same algorithm is also analyzed, which

computes the correspondences in the backward direction (from the last frame to the first frame) in addition to the forward direction. This method cannot handle occlusions, entries or exits. Salari and Sethi [SS90] handle these problems by first establishing correspondence for the detected points and then extending the tracking of the missing objects by adding a number of hypothetical points. Rangarajan and Shah [RS91] propose a greedy approach, which is constrained by proximal uniformity. Initial correspondences are obtained by computing the gradient based optical flow vector in the first two frames. The method does not address entry and exit of objects. If the number of detected points decreases occlusion or misdetection is assumed. Occlusion is handled by establishing the correspondence for the detected objects in the current frame. For the remaining objects, the position is predicted based on a constant velocity assumption. In the work by Intille et al. [IDB97], which uses a slightly modified version of [RS91] for matching centroids of objects, the objects are detected using background subtraction. The authors explicitly handle the change in number of objects by examining specific regions in the image, for example, a door, to detect entries/exits before computing the correspondence.

Veenman et al. [VRB01] extend the work of [SJ87, RS91] by using common motion constraints for correspondence. The algorithm is initialized by generating the initial tracks using a two-pass algorithm, and the cost function is minimized by the Hungarian assignment algorithm in two consecutive frames. This approach can handle occlusion and misdetection errors, however, it is assumed that the number of objects is the same throughout the sequence, i.e., no object enters or exits. See Figure ??a for tracking results.

Shafique and Shah [SS03] propose a multi-frame approach to preserve temporal coherency of the speed and position (Figure 2.7c). They represent the correspondence as a graph theoretic problem. Multiple frame correspondence relates to finding the best unique path $P_i = \{\mathbf{x}^0, \dots, \mathbf{x}^k\}$ for each point (the superscript represents the frame number). For mis-detected or occluded objects, the path will consist of missing positions in corresponding frames. The directed graph, which is generated using the points in k frames, is converted to a bipartite graph by splitting each node (object) into two (+ and -) nodes and representing directed edges as undirected edges from + to - nodes. The correspondence is then established by a greedy algorithm.

2.4.1.2 Statistical Methods for Correspondence

Measurements obtained from sensors invariably contain noise. Moreover, the object motions can undergo random perturbations. The statistical correspondence methods use the state space approach to model the object properties such as position, velocity and acceleration. These methods treat the tracking problem as inferring the object state by taking the measurement and the model uncertainties into account. Measurements usually consist of the object position in the image, which is obtained by a detection mechanism. Below, we will discuss the probabilistic state estimation methods in the context of point tracking, however it should be noted that these methods can be used in general to estimate the state of any time varying system. For example, these methods have extensively been used for tracking

contours [IB98], activity recognition [VRC03], object identification [ZCM03] and structure from motion [MSK89].

Consider a moving object in the scene. The information representing the object, e.g., location, is defined by a sequence of states $X^t : t = 1, 2, \dots$. The change in state over time is governed by the dynamic equation,

$$X^t = f^t(X^{t-1}) + W^t \quad (2.1)$$

where $W^t : t = 1, 2, \dots$ is white noise. The relationship between the measurement and the state is specified by the measurement equation $Z^t = h^t(X^t, N^t)$, where N^t is white noise and is independent of W^t . The objective of tracking is to estimate the state X^t given all the measurements up to that moment, or equivalently, to construct the probability density function $p(X^t | Z^{1, \dots, t})$. The theoretically optimal solution is provided by the recursive Bayesian filter which solves the problem in two steps. The *prediction* step uses the dynamic equation and the already computed PDF of the state at time $t - 1$ to derive the prior PDF of the current state, i.e., $p(X^t | Z^{1, \dots, t-1})$. Then, the *correction* step employs the likelihood function $p(Z^t | X^t)$ of the current measurement to compute the posterior PDF $p(X^t | Z^{1, \dots, t})$. If there is only one object in the scene, then the state can be simply estimated by the two steps defined hitherto. On the other hand, if there are multiple objects in the scene then measurements need to be associated with the corresponding object states. We discuss the two cases in the following.

Single Object State Estimation: For the single object case, if f^t and h^t are linear functions, and the initial state X^1 and noise have a Gaussian distribution then the optimal state estimate is given by the Kalman Filter. In the general case, i.e., if the state is not assumed to be a Gaussian, then state estimation can be performed using particle filters [Tan87].

- **Kalman Filters:** A Kalman filter can be used to estimate the state of a linear system where the state is assumed to be distributed as a Gaussian. Kalman filtering is composed of two steps: prediction and correction. The prediction step uses the state model to predict the new state of the variables:

$$\begin{aligned}\bar{X}^t &= \mathbf{D}X^{t-1} + W \\ \bar{\Sigma}^t &= \mathbf{D}\Sigma^{t-1}\mathbf{D}^T + Q^t\end{aligned}$$

where \bar{X}^t and $\bar{\Sigma}^t$ are the state and the covariance predictions at time t . \mathbf{D} is the state transition matrix which defines the relation between the state variables at time t and $t-1$. Q is the covariance of the noise W . Similarly, the correction step uses the current observations Z^t to update the object's state:

$$K^t = \bar{\Sigma}^t \mathbf{M}^T [\mathbf{M} \bar{\Sigma}^t \mathbf{M}^T + R^t]^{-1} \tag{2.2}$$

$$X^t = \bar{X}^t + K^t \underbrace{[Z^t - \mathbf{M}\bar{X}^t]}_v \tag{2.3}$$

$$\Sigma^t = \bar{\Sigma}^t - K^t \mathbf{M} \bar{\Sigma}^t,$$

where v is called the innovation, \mathbf{M} is the measurement matrix, K is the Kalman gain, which is the Riccati equation (2.2) used for propagation of the state models. Note that

the updated state, X^t , is still a Gaussian (see Fig. 2.9). If the functions f^t and h^t are non-linear, they can be linearized using the Taylor series expansion to obtain the Extended Kalman Filter [BF88]. The state is again assumed to be Gaussian.

The Kalman filter has been extensively used in the vision community for tracking. Broida and Chellappa [BC86] used Kalman filters to track points in noisy images. In stereo camera based object tracking, Beymer and Konolige [BK99] use a Kalman filter for predicting the object's position and speed in $x - z$ dimensions. Rosales and Sclaroff [RS99] use the extended Kalman filter to estimate the 3D trajectory of an object from 2D motion. A Matlab toolbox for Kalman filtering is available at [kal].

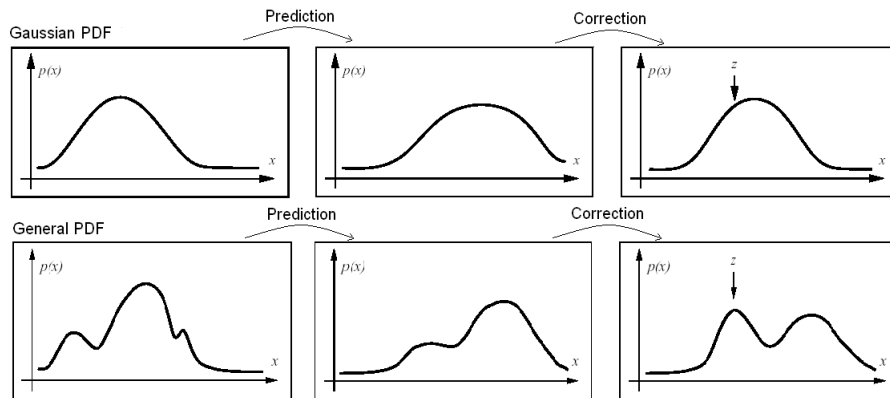


Figure 2.9: Row 1: A Kalman filter gives an optimal estimate of the posterior density given a Gaussian prior, and measurement densities and assuming linear dynamics. The posterior density is a Gaussian. Row 2: Particle filters can be used to propagate a general probability density function.

- **Particle Filters:** One limitation of the Kalman filter is the assumption that the state variables are normally distributed (Gaussian). Thus modeling state variables that do not have Gaussian distributions with the Kalman filters will result in poor state estimations. This limitation can be overcome by using particle filtering [Tan87]. In particle filtering, the conditional state density $p(X_t|Z_t)$ at time t is represented by a set of samples $\{s_t^{(n)} : n = 1, \dots, N\}$ (particles) with weights $\pi_t^{(n)}$ (sampling probability). The weights define the importance of a sample, i.e., its observation frequency [IB98]. To decrease computational complexity, for each tuple $(s_t^{(n)}, \pi_t^{(n)})$ a cumulative weight $c_t^{(n)}$ is also stored, where $c_t^{(N)} = 1$. The new samples at time t are drawn from $\mathbf{S}_{t-1} = \{(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}) : n = 1, \dots, N\}$ at the previous time $t - 1$ step based on different sampling schemes [Mac98]. The most common sampling scheme is “importance sampling” which can be stated as follows:

1. Selection: Select N random samples $\hat{s}_t^{(n)}$ from \mathbf{S}_{t-1} by generating a random number $r \in [0, 1]$, finding the smallest j such that $c_{t-1}^{(j)} > r$ and setting $\hat{s}_t^{(n)} = s_{t-1}^{(j)}$. Note that a sample can be selected multiple times.
2. Prediction: For each selected sample $\hat{s}_t^{(n)}$, generate a new sample by $s_t^{(n)} = f(\hat{s}_t^{(n)}, W_t^{(n)})$, where $W_t^{(n)}$ is a zero mean Gaussian error and f is a non-negative function, i.e., $f(s) = s$.
3. Correction: Weights $\pi_t^{(n)}$ corresponding to the new samples $s_t^{(n)}$ are computed using the measurements z_t by $\pi_t^{(n)} = p(z_t|x_t = s_t^{(n)})$, where $p(\cdot)$ can be modeled

as a Gaussian density, where $\sum_{n=1}^N \pi_t^{(n)} = 1$. Compute cumulative probabilities by $c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)}$.

Using the new samples \mathbf{S}_t , one can estimate the new position of the object by $\varepsilon_t = \sum_{n=1}^N \pi_t^{(n)} f(s_t^{(n)}, W)$. Particle filter based trackers can be initialized by either using the first measurements, $s_0^{(n)} \sim X_0$, with weight $\pi_0^{(n)} = \frac{1}{N}$ or by training the system using sample sequences. In addition to keeping track of the best particles, an additional resampling is usually employed to eliminate samples with very low weights. Note that the posterior density does not have to be a Gaussian (see Fig 2.9). Isard and Blake [IB98] use particle filters for contour tracking. A Matlab toolbox for tracking using particle filtering is available at [par].

Note that the Kalman and particle filter described above assume a single measurement at each time instant, i.e., the state of single object is estimated. Tracking multiple objects requires a joint solution of data association and state estimation problems.

Multi-Object Data Association and State Estimation: When tracking multiple objects using Kalman or particle filters, one needs to deterministically associate the most likely measurement for a particular object to that object's state, i.e., the correspondence problem needs to be solved before these filters can be applied. The simplest method to perform correspondence is to use the nearest neighbor approach. However, if the objects are close to each other, then there is always a chance that the correspondence is incorrect. An incorrectly associated measurement can cause the filter to fail to converge. There exist several statistical

data association techniques to tackle this problem. A detailed review of these techniques can be found in the book by Fortmann and Bar-Shalom [BF88] or in the survey by Cox [Cox93]. Joint Probability Data Association Filtering (JPDAF) and Multiple Hypothesis Tracking (MHT) are two widely used techniques for data association. We give a brief description of these techniques in the following.

- **Joint Probability Data Association Filter:** Let a track be defined as a sequence of measurements that are assumed to originate from the same object. Suppose we have N tracks and at time t , $Z(t) = z_1(t), \dots, z_{m_t}(t)$ are the m measurements. We need to assign these measurements to the existing tracks. Let η be a set of assignments. It is assumed that the number of tracks will remain constant over time. Let $v_{i,l}$ be the innovation (see the discussion on the Kalman Filter) associated with track l due to the measurement z_i . The JPDAF associates all measurements with each track. The combined weighted innovation is given by

$$v^l = \sum_{i=1}^{m_k} \beta_i^l v_{i,l}, \quad (2.4)$$

where β_i^l is the posterior probability that measurement i originated from the object associated with track l and is given as:

$$\beta_i^l = \sum_{\eta} P[\eta_l(k) | Z^t] \tau_{i,l}(\eta), \quad (2.5)$$

where $\tau_{i,l}$ is the indicator variable, with $i = 1, \dots, m_k$ and $l = 1, \dots, N$. τ is equal to one if measurement $z_i(k)$ is associated with track l , otherwise it is zero. The weighted

innovation given in (2.4) can be plugged in the Kalman filter update equations (2.3) for each track l .

JPDAF is used by Chang and Aggarwal [CA91] to perform 3D structure reconstruction from an video sequence. Rasmussen and Hager [RH01] use a constrained JPDAF filter to track regions. The major limitation of the JPDAF algorithm is its inability to handle new objects entering the field of view (FOV) or already tracked objects exiting the FOV. Since the JPDAF algorithm performs data association of a fixed number of objects being tracked over two frames, serious errors can arise if there is a change in the number of objects. The MHT algorithm, which is explained next, does not have this shortcoming.

- ***Multiple Hypothesis Tracking (MHT)***: If motion correspondence is established using only two frames, there is always a finite chance of an incorrect correspondence. Better tracking results can be obtained if the correspondence decision is deferred until several frames have been examined. The MHT algorithm maintains several correspondence hypotheses for each object at each time frame [Rei79]. The final track of the object is the most likely set of correspondences over the time period of its observation. The algorithm has the ability to create new tracks for objects entering the FOV and terminate tracks for objects exiting the FOV. It can also handle occlusions, i.e., continuation of a track even if some of the measurements from an object are missing. MHT is an iterative algorithm. An iteration begins with a set of current track hypotheses. Each hypothesis is a collection of disjoint tracks. For each hypothesis, prediction

of each object's position is made for the next frame, then the predictions are compared with actual measurements by evaluating a distance measure. A set of correspondences (associations) are established for each hypothesis based on the distance measure, which introduces new hypotheses for the next iteration. Each new hypothesis represents a new set of tracks based on the current measurements. Note that each measurement can belong to a new object entering the FOV, a previously tracked object, or a spurious measurement. Moreover, a measurement may not be assigned to an object because the object may have exited the FOV, or a measurement corresponding to an object may not be obtained. The latter happens when the object is occluded or it is not detected due to noise.

Note that MHT makes associations in a deterministic sense and exhaustively enumerates all possible associations. To reduce the computational load, Streit and Luginbuhl [SL94] propose a probabilistic MHT (PMHT) in which the associations are considered to be statistically independent random variables. Thus there is no requirement for exhaustive enumeration of associations. Recently, particle filters that handle multiple measurements to track multiple objects have been proposed by Hue et al. [HCP02]. In this method the data association is handled in similar manner to PMHT, but the state estimation is achieved through particle filters.

The MHT algorithm is computationally exponential both in memory and time. To overcome this limitation, Cox and Hingorani [CH96] use Murty's [Mur68] algorithm to determine the k-best hypotheses in polynomial time for tracking interest-points. Cham

and Rehg [CR99] use the multiple hypothesis framework to track the complete human body.

2.4.1.3 Discussion

In order to tackle noisy or missing observations, deterministic point trackers use heuristic constraints, i.e., common motion [VRB01] or proximal uniformity [RS91]. These methods usually use a greedy approach to establish the correspondence by minimizing a cost function. Thus the solution is not necessarily optimal. Statistical point tracking methods explicitly model the measurement and model uncertainties for tracking. These uncertainties are usually assumed to be in the form of normally distributed noise. However, the assumption that measurements are normally distributed around their predicted position may not hold. Moreover, in many cases the noise parameters are not known. In the case of valid assumptions on distributions and noise, Kalman filters [BF88] and MHT [Rei79] give optimal solutions.

2.4.1.4 Evaluation

The quantitative evaluation of point trackers can be performed by computing *precision* and *recall* measures. In this context, precision is the ratio of the number of correct correspondences to the number of total correspondences established. Recall is the ratio of the number of correct correspondences over the number of actual correspondences (ground truth). Point

trackers can be qualitatively compared on the basis of their ability to deal with entries of new objects, exits of objects, missing observations (occlusion) and to provide optimal solutions for correspondences. In Table 2.3, we provide a comparison based on these properties.

Table 2.3: Qualitative comparison of point trackers. #: number of objects, **M**: multiple objects, **S**: single object, \checkmark : available, \times : not available.

	#	Entry	Exit	Occlusion	Optimal
GE [SJ87]	M	\times	\times	\times	\times
MGE [SS90]	M	\checkmark	\checkmark	\checkmark	\times
GOA [VRB01]	M	\times	\times	\checkmark	\checkmark
MFT [SS03]	M	\checkmark	\checkmark	\checkmark	\times
Kalman tracker [BF88]	S	\times	\times	\times	\checkmark
JPDAF tracker [BF88]	M	\times	\times	\times	\times
MHT [CH96]	M	\checkmark	\checkmark	\checkmark	\checkmark

2.4.2 Tracking by Matching Primitive Geometric Regions

Given the object in the first frame, these trackers compute the parametric motion, e.g., translation, translation+rotation and affine transformation, of the object in subsequent frames. These algorithms differ in terms of the appearance representation used, the number of objects

tracked and the method used to estimate the motion parameters. We divide these tracking methods into two subcategories based on the appearance representation used, namely templates and density based appearance models, and multi-view appearance models.

2.4.2.1 Tracking using Template and Density Based Appearance Models

Templates and density based appearance models (see Section 2.1) have been widely used in the area of tracking. In general, these trackers model the object and its motion independently of the other objects in the scene. However, in the case of multi-object tracking, the independence assumption may not be valid. We divide the trackers in this category into two sub-categories based on whether the objects are tracked individually or jointly.

Tracking Single Objects The most common approach in this category is “template matching”. Template matching is a brute force method of searching the image, I_w , for a region similar to the object template, O_t . The position of the template in the current image is

computed by a similarity measure, e.g., cross correlation: $\arg \max_{dx, dy} \frac{\sum_x \sum_y (O_t(x, y) \times I_w(x+dx, y+dy))}{\sqrt{\sum_x \sum_y O_t^2(x, y)}}$

where (dx, dy) specifies the candidate template position. Usually image intensities or color features are used to form the templates. Since image intensity is very sensitive to illumination changes, image gradients [Bir98] can also be used as features. A limitation of template matching is high computation cost due to the brute force search. To reduce the computational cost, researchers usually limit the object search to the vicinity of its previous position.

Note that, instead of templates, other object representations can also be used for tracking, for instance, color histograms or mixture models can be computed by using the pixels inside the rectangular or ellipsoidal regions. Fieguth and Terzopoulos [FT97] generate object models by finding the mean color of the pixels inside the rectangular object region. To reduce computational complexity, they search for the object in eight neighboring locations. The similarity between the object model, M , and the hypothesized position, H , is computed by evaluating the ratio between the color means computed from M and H . The position that provides the highest ratio is selected as the current object location.

Comaniciu and Meer [CRM03] use a weighted histogram computed from an elliptical region to represent the object. Instead of performing a brute force search for locating the object, they use the mean shift procedure (2.3.3). The mean shift tracker maximizes the appearance similarity iteratively by comparing the histograms of the object, Q , and the window around the hypothesized object location, P . The histogram similarity is defined in terms of the Bhattacharya coefficient: $\sum_{u=1}^b P(u)Q(u)$, where b is the number of bins. At each iteration, the mean shift vector is computed such that the histogram similarity is increased. This process is repeated until convergence is achieved, which usually takes five to six iterations. For histogram generation, the authors use a weighting scheme defined by a spatial kernel which gives higher weights to the pixels closer to the object center. Comaniciu [Com02] extended the mean shift tracking approach by representing the object using a joint spatial-color histogram (Sec. 2.3.3). An example of mean shift tracking is given in Figure 2.10.

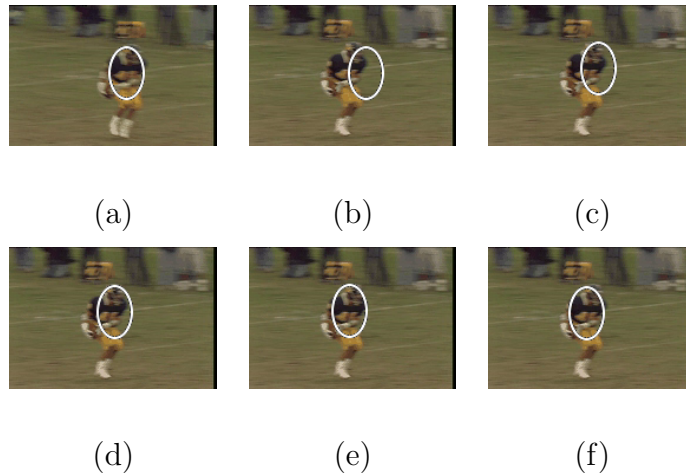


Figure 2.10: Mean shift tracking iterations. (a) Estimated object location at time $t - 1$, (b) Frame at time t with initial location estimate using the previous object position, (c), (d), (e) location update using mean shift iterations, (f) final object position at time t .

An obvious advantage of the mean shift tracker over the standard template matching is the elimination of a brute force search, and the computation of the translation of the object patch in a small number of iterations. However, mean shift tracking requires that a portion of the object is inside the elliptical region upon initialization (the object has to be inside the white ellipse in Figure 2.10b). Implementation of the mean shift tracker is available in OpenCV as CAMSHIFT at [meab].

Shi and Tomasi, in [ST94], propose the KLT tracker based on the principle that only good features can be tracked well. Their approach consists of two steps. The first step finds the translation of an interest point and the second step monitors the quality of each interest point. Given a set of interest points computed using the KLT detector discussed

in Sec. 2.3.1, the translation (u, v) of the patch centered on the point is iteratively computed by $u \leftarrow u + du$ and $v \leftarrow v + dv$ where the updates du and dv are obtained from

$$\begin{pmatrix} \sum \sum I_x^2 & \sum \sum I_x I_y \\ \sum \sum I_x I_y & \sum \sum I_y^2 \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix} = \begin{pmatrix} \sum \sum I_x I_t \\ \sum \sum I_y I_t \end{pmatrix}$$

similar to the optical flow computation proposed by Lucas and Kanade [LK81]. Once the new location of the interest point is obtained using the computed (u, v) , in the second step, the authors compute the affine transformation between the corresponding patches in consecutive frames. If the sum of square differences between the current patch and the projected patch is small, they continue tracking the feature, otherwise the feature is eliminated. The implementation of the KLT tracker is available at [klt].

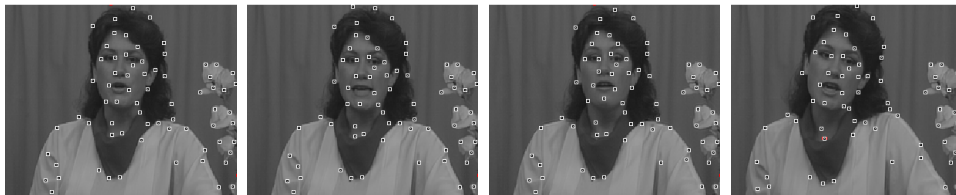


Figure 2.11: Tracking features using the KLT tracker.

Jepson et al. [JFE03] propose an object tracker that tracks an object as a three component mixture, consisting of the stable appearance features, transient features and noise process. The stable component identifies the most reliable appearance for motion estimation, i.e., the regions of the object whose appearance does not quickly change over time. The transient component identifies the quickly changing pixels. The noise component handles the outliers in the object appearance, which arise due to noise. An online version of the EM

algorithm is used to learn the parameters of this three component mixture. The authors use the phase of the steerable filter responses as features for appearance representation. The object shape is represented by an ellipse. The motion of the object is calculated in terms of warping the tracked region from one frame to the next one. The warping transformation consists of translation, rotation and scale parameters. A weighted combination of the stable and transient components is used to determine the warping parameters. The advantage of learning stable and transient features is that one can give more weight to stable features for tracking, for example, if the face of a person who is talking is being tracked, then the forehead or nose region can provide a better match to the face in the next frame as opposed to the mouth of the person.

Tracking Multiple Objects Modeling objects individually does not take into account the interaction between multiple objects and between objects and background during the course of tracking. An example interaction between objects can be one object partially or completely occluding the other. The tracking methods given below model the complete image, i.e., the background and all moving objects are explicitly tracked.

Tao et al. [TSK02] propose an object tracking method based on modeling the whole image, I^t , as a set of layers. The representation includes a single background layer and one layer for each object. Each layer consists of a shape prior (ellipse), Φ , motion model (translation and rotation), Θ , and layer appearance (intensity modeled using a single Gaussian), A . A layering is performed by first compensating the background motion modeled by projective

motion, such that the object’s motion can be estimated from the compensated image using 2D parametric motion. Then, each pixel’s probability of belonging to a layer (object), p_l , is computed based on the object’s previous motion and shape characteristics. Any pixel far from a layer is assigned a uniform background probability, p_b . Later, the object’s appearance (intensity, color) probability p_a is coupled with p_l to obtain the final layer estimate. The model parameters (Φ_t, Θ_t, A_t) that maximize observing a layer at time t are estimated iteratively using an expectation maximization algorithm. However, due to the difficulty in simultaneously estimating the parameters, the authors individually estimate one set while fixing the others. For instance, they first estimate layer ownership using intensity for each pixel, then they estimate the motion (rotation and translation) using appearance probabilities and finally update layer ownership using this motion. The unknowns for each object are iteratively estimated until the layer ownership probabilities are maximized.

Isard and MacCormick [IM01] propose joint modeling of the background and foreground regions for tracking. The background appearance is represented by a mixture of Gaussians for the background over small patches. The appearance of all foreground objects is modeled by a mixture of Gaussians. The shape of objects is modeled as cylinders. They assume the ground plane is known, thus the 3D object positions can be computed. Tracking is achieved by using particle filters, where the state vector includes the 3D position, shape and velocity of all objects in the scene. They propose a modified prediction and correction scheme for particle filtering, which can increase or decrease the size of the state vector to include or remove objects. The method can also tolerate occlusion between objects. However, the

maximum number of objects in the scene is required to be predefined. Another limitation of the approach is the use of same appearance model for all foreground objects and it requires training to model the foreground regions.

2.4.2.2 Tracking Using Multi-view Appearance Models

In the aforementioned tracking methods, the appearance models, like histograms and templates are usually generated on line. Thus these models represent the information gathered about the object from the most recent observations. The objects may appear different from different views and if the object view changes dramatically during tracking, the appearance model may no longer be valid and the track of the object might be lost. To overcome this problem, different views of the object can be learned off-line and used for tracking.

In [BJ98], Black and Jepson propose a subspace based approach, i.e., Eigenspace, to compute the affine transformation from the current image of the object to the image reconstructed using eigenvectors. First, a subspace representation of the appearance of an object is built using Principal Component Analysis (PCA), then the transformation from the image to the Eigenspace is computed by minimizing the so-called subspace constancy equation, which evaluates the difference between the image reconstructed using the eigenvectors and the input image. Minimization is performed in two steps: finding subspace coefficients and computing affine parameters. In the first step, the affine parameters are fixed and the subspace coefficients are computed. In the second step, using the new subspace coefficients,

affine parameters are computed. Based on this, tracking is performed by estimating the affine parameters iteratively until the difference between the input image and the projected image is minimized.

In a similar vein, Avidan [Avi01] uses a State Vector Machine (SVM) classifier for tracking. SVM is a general classification scheme that, given a set of positive and negative training examples, finds the best separating hyperplane between the two classes [Vap98]. During testing, the SVM gives a score to the test data indicating the degree of membership of the test data to the positive class. For SVM based trackers, the positive examples consist of the images of the object to be tracked and the negative examples of all other things that are not to be tracked. Generally negative examples consist of background regions that could be confused with the object. Avidan's tracking method, instead of trying to minimize the intensity difference of a template from image regions, tries to maximize the SVM classification score over image regions in order to determine the location of the object.

2.4.2.3 Discussion

Trackers in this category use primitive geometric models for shape representation. Primitive geometric shape models are suitable representations for rigid objects, though they have also been used in the context of tracking non rigid objects. These tracking methods model the appearance by templates, probability densities or multi-view models. The motion is defined in terms of translation, translation+rotation, affine or projective transformation.

The methods that use gradient descent based minimization for motion estimation require overlap between object regions in successive frames [CRM03, ST94, Avi01], however this assumption is valid for many tracking scenarios.

2.4.2.4 Evaluation

The evaluation criteria for these trackers depend on the context in which the tracking algorithm is being used. The goal of the tracking algorithm can be either to:

1. estimate the motion of an object, or
2. to estimate the region encompassed by an object

in every frame. For the former goal, the evaluation can be performed by computing a distance measure between the estimated and actual motion parameters. An example of a distance measure can be the angular distance, $d = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| |\mathbf{B}|}$, between the motion vectors, \mathbf{A} and \mathbf{B} . For region based evaluation, precision and recall measures can be used. In this context, both of these quantities are defined in terms of the intersection of the hypothesized and correct object regions. In particular, precision is the ratio of the intersection to the hypothesized region. The recall is the ratio of the intersection to the ground truth.

Trackers using geometric shape models can be qualitatively compared based on

- tracking single or multiple objects,

- ability to handle occlusion,
- requirement for training,
- type of motion model, and
- requirement of manual initialization.

In Table 2.4, we qualitatively compare the methods discussed in this section.

2.4.3 Tracking by Contour Evolution

Many non-rigid objects have complex shapes which cannot be well described by simple geometric shape representations, e.g., hands, head and shoulders (see Fig 2.12a). Contour based object models provide an accurate shape description for these objects. The goal of a contour based object tracker is to find the boundary between the object and the background in each frame, such that the object region is tightly enclosed within the contour. We can categorize the contour based object trackers into two categories based on how the contour is evolved. The first category of contour trackers use state space models (Sec. 2.4.1.2) to evolve the contour. The second category performs direct minimization of a contour energy functional.

Table 2.4: Qualitative comparison of geometric model based trackers. “Init.” denotes initialization. #: number of objects, **M**: multiple objects, **S**: single object respectively, **A**: affine homography, **T**: translational motion, **S**: scaling, **R**: rotation, **P**: partial occlusion, **F**: full occlusion.

	#	Motion	Training	Occlusion	Init.
Simple template matching	S	T	×	P	✓
Mean shift [CRM03]	S	T+S	×	P	✓
KLT [ST94]	S	A	×	P	✓
Appearance Tracking [JFE03]	S	T+S+R	×	P	✓
Layering [TSK02]	M	T+S+R	×	F	×
Bramble [IM01]	M	T+S+R	✓	F	×
EigenTracker [BJ98]	S	A	✓	P	✓
SVM [Avi01]	S	T	✓	P	✓

2.4.3.1 Tracking Using State Space Models

In this section, we will discuss tracking object contours that use the probabilistic state space approaches to track the object. The object’s state is defined in terms of the shape and the motion parameters of the contour. The state is updated at each time instant such that the contour’s *a posteriori* probability is maximized. The posterior probability depends on the

prior state and the current likelihood, which is usually defined in terms of the distance of the contour from observed edges.

Terzopoulos and Szeliski [TS92] define the object state by the dynamics of the control points. The dynamics of the control points are modeled in terms of a *spring model*, which moves the control points based on the spring stiffness parameters. The new state (spring parameters) of the contour is predicted using Kalman filters. The correction step uses the image observations which are defined in terms of the image gradients.

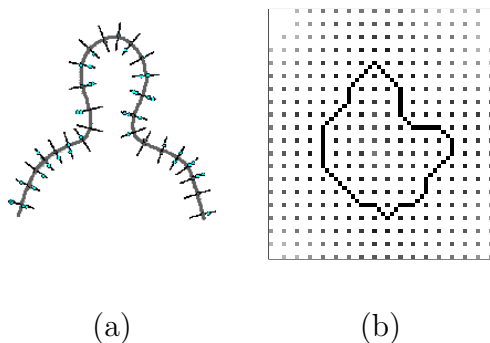


Figure 2.12: (a) Edge observations along the contour normals.(b) Level-set contour representation; each grid position encodes the Euclidean distance from the contour, gray level represents the value of the grid.

In [IB98], Isard and Blake define the object state in terms of spline shape parameters and affine motion parameters. The measurements consist of image edges computed in the normal direction to the contour (see Figure 2.12a). The state is updated using a particle filter. In order to obtain initial samples for the filter, they compute the state variables from

the contours extracted in consecutive frames during a training phase. During the testing phase, the current state variables are estimated through particle filtering based on the edge observations along normal lines at the control points on the contour.

In [MB00], MacCormick and Blake extend the particle filter based object tracker in [IB98] to track multiple objects by including the “exclusion principle” for handling occlusion. The exclusion principle integrates into the sampling step of the particle filtering framework, such that for two objects, if a feature is lying in the observation space of both objects then it contributes more to the samples of the object which is occluding the other object. Since the exclusion principle is only defined between two objects, this approach can track at most two objects undergoing occlusion at any time instant.

Chen et al. [CRH01] propose a contour tracker, where the contour is parameterized as an ellipse. Each contour node has an associated HMM and the state of each HMM is defined by the points lying on the lines normal to the contour control point. The observation likelihood of the contour depends on the background and the foreground partitions defined by the edge along the normal line on contour control points. The state transition probabilities of the HMM are estimated using the JPDAF. Given the observation likelihood and the state transition probabilities the current contour state is estimated using the Viterbi algorithm [Vit67]. After the contour is approximated, an ellipse is fit to enforce the elliptical shape constraint.

The methods discussed above represent the contours using explicit representation, e.g., parametric spline. Explicit representations do not allow topology changes, such as region

split or merge [Set99]. Next, we will discuss contour tracking methods based on direct minimization of an energy functional. These methods can use implicit representations and allow topology changes.

2.4.3.2 Tracking By Direct Minimization of Contour Energy Functional

In terms of evolving the contour onto the object region, there is an analogy between the segmentation methods discussed in Sec. 2.3.3 and the contour tracking methods in this category. Both the segmentation and tracking methods minimize the energy functional either by greedy methods or by gradient descent. The contour energy is computed using temporal information in the form of either the temporal gradient (optical flow) [BSR00, Man02, CS03], or appearance models computed from object and background regions [Ron94].

Contour tracking using temporal image gradients is motivated by the extensive work on computing the optical flow. The optical flow constraint is derived from the brightness constancy constraint: $I^{t+1}(x, y) - I^t(x - u, y - v) = 0$, where I is the image, t is the time, and (u, v) is the flow vector in the x and the y directions. Bertalmio et al. [BSR00] use this constraint to evolve the contour in consecutive frames. Their objective was to compute u and v iteratively for each contour position using the level set representation (see Figure 2.12b). At each iteration, contour speed in the normal direction, \vec{n} , is computed by projecting the gradient magnitude $|\nabla I_t|$ on \vec{n} . The authors use two energy functionals, one for contour tracking, E_t , and one for intensity morphing, E_m : $E_m(\Gamma) = \int_0^1 E_{im}(\mathbf{v})ds$ and $E_t(\Gamma) =$

$\int_0^1 E_{ext}(\mathbf{v})ds$, where E_{ext} is computed based on E_m . The intensity morphing functional, which minimizes intensity changes in the current and the previous frames, $\nabla I_t = I_t - I_{t-1}$, on the hypothesized object contour: $\frac{\partial F(x,y)}{\partial t} = \nabla I_t(x,y) \parallel \nabla F(x,y) \parallel$, is coupled with the contour tracking equation: $\frac{\partial \phi(x,y)}{\partial t} = \nabla I_t(x,y) \vec{n}_F \vec{n}_\phi \parallel \nabla \phi(x,y) \parallel$, and both functionals are minimized simultaneously. For instance, if $\nabla I_t(x,y) \gg 0$, then the contour moves with the maximum speed in its normal direction and $I^{t-1}(x,y)$ is morphed into $I^t(x,y)$. On the other hand, if $\nabla I_t(x,y) \approx 0$, then the evolution speed will be zero.

Similarly, Mansouri [Man02] uses the optical flow constraint for contour tracking. In contrast to [BSR00] which computes the flow only on the object boundary, his approach is motivated by computing the flow vector for each pixel inside the complete object region in a circular neighborhood with radius r using a brute-force-search. Once the flow vectors are computed, the contour energy, which is based on the brightness constancy constraint, is evaluated. This process is iteratively performed until the energy is minimized.

In [CS03], Cremers and Schnorr use the optical flow as an feature for contour evolution, such that an object can only have homogeneous flow vectors inside the region. Their energy is a modified form of the common Mumford-Shah energy [MS89], which evolves the contour until a region with homogeneous flow vectors is encountered. They also incorporated the shape priors to better estimate the object shape. The shape priors are generated from a set of object contours, such that each control point on the contour has an associated Gaussian with a mean of the spatial positions of the corresponding control points on all the contours along with a standard deviation.

Tracking methods using region priors do not explicitly use the temporal information in the form of the brightness constancy constraint. In contrast, they use appearance priors generated on line. Tracking is performed by initializing the contour in the current frame with its previous position. Ronfrad [Ron94] propose a region based contour tracking method. His functional is defined based on the piecewise stationary image models formulated as Ward distances, which is a measure of image contrast [BG89]. Since the resulting energy does not have an analytical form, each contour point is evolved individually based on its local neighborhood.

2.4.3.3 Discussion

The most important advantage of a contour tracker is that it can model a large variety of object shapes. Contours are represented by explicit (control points and splines) or implicit (level sets) representations (see Figure 2.12). The use of these representations depend on the context of the application. For instance, in most cases for tracked objects that do not split or merge, explicit representation is usually suitable for tracking. However, in some applications such as surveillance, it is important to keep track of a person leaving an object behind. In the context of contour tracking, when a person leaves an object, part of the object contour will be placed on the left object (region split). Topology changes like region split or merge can be handled well by implicit representations.

2.4.3.4 Evaluation

Contour trackers are employed when tracking the complete region of an object is required. In the context of region tracking, the precision and recall measures are defined in terms of the intersection of the hypothesized and correct object regions. The precision is the ratio of the intersection to the hypothesized region and recall is the ratio of the intersection to the ground truth. Qualitatively, the contour based methods can be compared on the basis of requirement of training and occlusion handling. Moreover some algorithms only use information on the contour boundary for evolution while other use the complete region. Generally the region based approaches are more resilient to noise. A qualitative comparison of contour based approaches is given in Table 2.5.

2.5 Relevant Issues

The primary issues that arise in the course of development of any tracking system include the determination of an appropriate representation of the object and the selection of suitable image features. Although we already introduced these issues, we give a further discussion in light of the presented tracking algorithms. Further more, we give a brief discussion on the methods for occlusion resolution in tracking. Finally, we debate the need and use of multiple cameras for tracking objects.

Table 2.5: Qualitative comparison of contour evolution based trackers. #: number of objects,

S : single, **M**: multiple, **P**: partial, **F**: full, **B**: boundary, **R**: Region.

	#	Occlusion	Training	Region or Boundary
[TS92]	S	×	✓	B
[IB98]	S	×	✓	B
[MB00]	M	F	✓	B
[CLL01]	S	×	✓	B
[Man02]	S	×	×	R
[BSR00]	S	×	×	R
[PD02]	S	×	×	B
[CKS02]	S	P	✓	R

2.5.1 Object Representation

In order to represent the appearance of an object, models can be built online or can be defined a priori. The primitive geometric shape models are only useful for describing simple rigid objects. These models are usually defined a priori and are not general enough to handle a variety of different objects. Subspace approaches, e.g., Principal Component Analysis (PCA), Independent Component Analysis (ICA), have been used for both shape and appearance representation [MP97, BJ98]. Using eigenspace for similarity computation

is a useful alternative to standard template matching techniques such as SSD and normalized correlation. The Eigenspace based similarity computation is equivalent to matching with a linear combination of eigen templates. This allows for distortions in the templates, for example distortion caused by illumination changes in color images. Recently classifiers encoding multiple views, e.g., SVM, have also been used as object representation. These classifiers are learned using both positive and negative examples, and tracking is achieved by maximizing a classification score across image regions. One advantage of this approach over simple template matching is that knowledge about background objects (negative examples that are not to be tracked) is explicitly incorporated in the tracker. Both the subspace and classifier based representations require extensive a priori information about the objects to be tracked.

For exact tracking of non-rigid object regions, e.g., a hand or complete human body, a more detailed representation might be required. Contour based 2D object representations can naturally deal with the non-rigid object motion. Contours can be defined by using both implicit and explicit representations. In computer vision community most commonly used implicit representation is the level sets. Besides the level sets, volume fluid methods [Hir90], which have been well studied in the fluid dynamics community, can be used for contour based tracking.

One interesting approach for on line learning of non-rigid shape models was proposed by Bregler et al. [TB02]. They automatically estimate the object shape using a structure from motion approach instead of imposing predefined shape constraints, and use these constraints

to track points on the object. Initially, tracks of a set of interest-points on an object are obtained using an approach similar to the KLT tracker for the complete sequence. These tracks are used to estimate n basis-shapes of the object. These basis-shapes are used to compute hypothesis tracks for any point on the object. The confidence of each hypothesis is measured by computing the SSD between the patch in the first frame and the remaining frames. The hypothesis with the highest confidence is selected as the track of the point. A limitation of this approach is that tracking has to be performed off line for a set of interest points. In addition, the sequence has to be long to find accurate shape basis.

2.5.2 Feature Selection

Feature selection methods can be divided into *filter* methods and *wrapper* methods [BL97]. The filter methods try to select the features based on a general criteria, e.g., the features should be uncorrelated. The wrapper methods select the features based on the usefulness of the features to solve the given problem, e.g., the classification performance using a subset of features. The Principal Component Analysis (PCA) is an example of the filter methods for the feature reduction. PCA involves transformation of number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called the principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. A wrapper method of selecting the discriminatory features for tracking a particular class

of objects is the Adaboost [TV04] algorithm. One problem with off line feature selection approaches is that the feature selection and the combination depends greatly on the training examples. Also, training samples may not always be available in general tracking scenarios.

Collins and Liu. [RY03] compute the “best” features, given a pool of features, in every frame for object tracking. The criteria for goodness of a feature is that it should be *locally* discriminative, i.e., the object only needs to be distinguishable from its immediate surroundings. At each time instant, given the object location, a variance ratio (ratio of within class and between class likelihoods) for the object and the background classes is computed for each feature. The features with the highest variance ratio are then used for tracking.

Overall, if a priori object information is available, one can use the filter or wrapper methods for the feature selection from the training set. However, there still is a need for on line selection of features that are the most discriminative at a particular time.

2.5.3 Resolving Occlusion

Occlusion can be classified into three categories: self occlusion, inter-object occlusion and occlusion by the background scene structure. Self occlusion occurs when one part of the object occludes another. This situation most frequently arises while tracking articulated objects. Inter-object occlusion occurs when two objects being tracked occlude each other. Similarly, occlusion by the background occurs when a structure in the background occludes

the tracked objects. Generally, for inter-object occlusion, the multi-object trackers like [MB00, EDH02] can exploit the knowledge of position and appearance of the occluder and occludee to detect and resolve occlusion. Partial occlusion of an object by a scene structure is hard to detect, since it is difficult to differentiate between the object changing its shape and the object getting occluded.

A common approach to handle complete occlusion during tracking is to model the object motion by linear dynamic models or by non-linear dynamics and in case of occlusion, to keep on predicting the object location till the object re-appears. For example, a linear velocity model is used in [BK99] and a Kalman filter is used for estimating the location and motion of objects. A non-linear dynamic model is used in [IM01] and a particle filter was employed for state estimation.

Researchers have also utilized other features to resolve occlusion, e.g., silhouette projections [HHD00] (to locate persons' heads during partial occlusion), optical flow [DT01b] (assuming that two objects move in opposite directions). Free form object contour trackers employ a different occlusion resolution approach. These methods usually address occlusion by using shape priors which are either build ahead of time [CKS02] or build on line [YS04]. In particular Cremers et al. [CKS02] build a shape model from subspace analysis (PCA) of possible object shapes to fill in missing contour parts. Yilmaz et al. [YS04] build on-line shape priors using a mixture model based on the level set contour representation. The approach is able to handle complete object occlusion.

The chance of occlusion can be reduced by an appropriate selection of camera positions. For instance, if the cameras are mounted on airborne vehicles, i.e., a birds eye view of the scene is available, occlusions between objects on the ground do not occur. However, oblique view cameras are likely to encounter multiple object occlusions and require occlusion handling mechanisms. Multiple cameras viewing the same scene can also be used to resolve object occlusions during tracking [DT01a, MD03]. These methods are discussed in the next subsection.

2.5.4 Limitations of Single Camera and Multiple Camera Tracking

The need for using multiple cameras for tracking arises due to two reasons. The first reason is the use of depth information for tracking and occlusion resolution. The second reason for using multiple cameras is to increase the area under view since it is not possible for a single camera to observe large areas because of finite sensor resolution. An important issue in using multiple cameras is the requirement of knowledge of the relationship between the different camera views, which can be manually defined [CLF01, CA99] or can be computed automatically [LRS00, KS03] from the observations of the objects moving in the scene. For the tracking algorithms that require the depth estimation, high computational cost is another concern. Due to the availability of successful commercial products, real-time depth recovery is solved to some extent, and can be used as a feature for the existing tracking methods. In addition, methods like [MD03] do not perform dense depth estimation but calculate a single

depth estimate for each object using a region based stereo method, which also reduces the computational load. Multi-camera tracking methods like [DT01a, MD03] have demonstrated superior tracking results as compared to single camera trackers in case of persistent occlusion between the objects.

The aforementioned multi-camera tracking methods assume stationary cameras. Recently Kang et al. [KCM03] use a combination of stationary and pan-tilt-zoom cameras with overlapping views for tracking. In many situations it is not possible to have overlapping camera views due to limited resources or large areas of interest. Methods for tracking in such a scenario inherently have to deal with sparse object observations due to non-overlapping views. Therefore some assumptions have to be made about the object speed and the path, in order to obtain the correspondences across cameras [HR97, KZ99, JRS03]. Note that these methods, which establish object correspondence across non-overlapping cameras, assume 1) the cameras are stationary and 2) the object tracks within each camera are available. The performance of these algorithms depends greatly on how much the object follows the established paths and expected time intervals across cameras. For scenarios in which spatio-temporal constraints can not be used, e.g., objects moving arbitrarily in the non-overlap region, only “tracking by recognition” approach can be employed, which uses the appearance and the shape of the object to recognize it when reappears in a camera view.

2.6 Conclusions

In this chapter, we present an extensive survey of object tracking methods and also give a brief review of related topics. We divide the tracking methods into three categories based on the use of object representations: namely, methods establishing point correspondence, methods using primitive geometric models and methods using contour evolution. Note that all these classes require object detection at some point. For instance, the point trackers require detection in every frame, whereas geometric region or contours based trackers require detection only when the object first appears in the scene. Recognizing the importance of object detection for tracking systems, we include a short discussion on popular object detection methods. We provide detailed summaries of object trackers, including discussion on the object representations, motion models, and the parameter estimation schemes employed by the tracking algorithms. Moreover, we describe the context of use, degree of applicability, evaluation criteria and qualitative comparisons of the tracking algorithms. We believe that being the first survey in object tracking area with a rich bibliography content, can give adequate insight the readers into this important research topic and encourage new research.

CHAPTER 3

VISUAL FEATURES

Selecting the right features plays a critical role in tracking. In general, the most desirable property of a visual feature is its uniqueness, so that the objects can easily be distinguished in the feature space. In general, many tracking algorithms use a combination of these features. The most common features used during the last two decades are color (intensity) and texture. Color feature is primarily generated by the reflectance of the light from the object surface. The texture feature measures the variation of the color. Besides color and texture, edge and optical flow are also used to represent the objects. Edges are generated by the strong changes in image intensities. Similar to the texture feature, edges are less sensitive to illumination changes compared to the color feature. Optical flow represents a dense field of displacement vectors which defines the translation of each pixel in a region. It is from the brightness constraint, which assumes “brightness constancy” of corresponding pixels in the consecutive frames [HS81].

In this chapter, we discuss color and texture features in detail due to their relevance to the proposed tracking methods proposed in Chapters 4 and 5. We will also summarize the

approaches to model these features. In addition, we will discuss possible methods on fusing a subset of these features for increasing the robustness of methods.

3.1 Color

The apparent color of an object, which is the visible spectrum of light, is influenced primarily by the following physical factors:

- The spectral power distribution of the illuminant, and
- The surface reflectance properties of the object.

In digital images, the physical properties are mapped into a predefined color space composed of several bands, such as the three primary colors of light: Red (R), Green (G), and Blue (B). The RGB color space is highly correlated and is not a perceptually uniform color space, i.e., the difference between the colors in the RGB space does not correspond to the color differences perceived by the humans [Pas01]. There are various other spaces to represent the color. Common color spaces used in literature are RGB, HSI (hue, saturation, intensity), HSV (hue, saturation, value), YIQ (Y-axis, in-phase, quadrature) etc. These color spaces are usually sensitive to noise [SKP96]. There are also less common color spaces, which are tuned according to the application they are used in. For instance, RGS, which was used in [ED01], produces two chromaticity components, which are normalized red and green values and a lightness component, S, which represents the strength of illumination. Chromaticity

components have been used for various vision applications because of their property of being minimally effected by illumination variation, similar to the H and V components of HSV color space. RGS color model is simply computed by

$$r = \frac{R}{R + G + B}, \quad g = \frac{G}{R + G + B}, \quad s = \frac{R + G + B}{3}$$

The small letters for the RGS model components are used for distinguishing them from the RGB model. In Figure 3.1, we show the components of the RGS color space.



Figure 3.1: (a) RGB image. (b) r , (c) g , (d) s components of the RGS model.

3.1.1 Modeling Color

Modeling color aims at selecting the right representation of the color content of an image. In general, the performance of color based trackers degrades due to illumination variation. However, it has been shown that in practice color based approaches have reasonable performances [CRM00, CM99, YSL01]. Common approaches for color modeling are:

- *Object template*: contains raw color values in a rectangular patch. Object templates have been successfully used in tracking facial features [KP02], object tracking and recognition [Ols00], etc. However, they are only good for tracking for very short durations, and since color values may change, they perform poorly for longer durations [JFE01].
- *Histogram*: captures the distribution of color in an image or a patch. The color space is quantized into several discrete bins, and the number of pixels with the same color are recorded in the corresponding bin. Histogram is widely used for image segmentation, object recognition, object tracking etc.
- *Simple parametric model*: The data is usually modeled by a single kernel (filter). The most common model used in this context is the Gaussian kernel with two unknown parameters: mean and standard deviation. Data is used to compute the model unknowns.

- *Mixture models*: The most common parametric model is the mixture of Gaussians, whose parameters are computed using the Expectation Maximization (EM) algorithm. Mixture of Gaussians represent the data better than the single Gaussian models, however the computation cost of estimating model parameters is usually more expensive. In addition, number of components used to represent the data is hard to estimate.
- *Kernel density estimate*: is the general form of histograms, which produce continuous density estimate. Since there are no parameters to estimate it is the most efficient model. Usually, model is generated by storing all observations of the data, and assuming every individual observation is itself a distribution modeled by a simple parametric model such as Gaussian and Epanechnikov. For an input datum the probability that it comes from that kernel density estimate is computed by summing the probabilities computed from individually modeled observations. Kernel density estimation has been successfully used in object tracking [CRM00, YS02a, YSL01].

Depending on how the color is utilized, the resulting feature space can be variant to scene deformations (templates) and may not hold spatial relationships between colors (histograms, kernel density estimates). To introduce spatial relationship between colors, various researchers modeled both the spatial position and color using nonparametric models [KS01, HE02, CM99].

Among different approaches, in the context of tracking objects, we used the kernel density to model the color due to its flexibility. For instance, template based models are usually

prone to changes in shape or color of the object. The mixture models are expensive to compute, and the number of components used affects the quality of the model. In addition, it was demonstrated in [JR02] that the performance of histogram is better than a mixture model in context of skin color modeling. In contrast to histograms, kernel density estimates model non-uniformly distributed data continuously similar to a parametric model and is statistically a better choice [Fuk90].

Kernel density estimation can be computed using the multivariate kernel $K(\mathbf{x})$ by:

$$\tilde{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K(\mathbf{x} - \mathbf{x}_i),$$

where d is the dimension of data (for RGS space $d = 3$ and for gray level images $d = 1$), and h is the bandwidth of the kernel. If the kernel is chosen to be a uniform kernel, density estimation will reduce to a histogram, which is a discrete density estimate. To obtain continuous density estimates other kernels can be used including: Gaussian, Triangular, Bi-weight or Epanechnikov. The Epanechnikov kernel is given by:

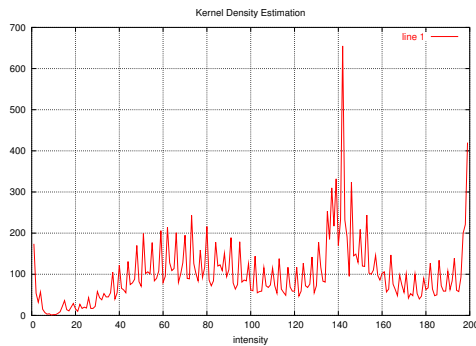
$$K(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2) & \text{if } \|\mathbf{x}\| < 1 \\ 0 & \text{otherwise} \end{cases},$$

where c_d is the volume of unit d -dimensional sphere. Among various kernels, this kernel yields minimum average global error between the estimate and the true density [CRM00, Sco92].

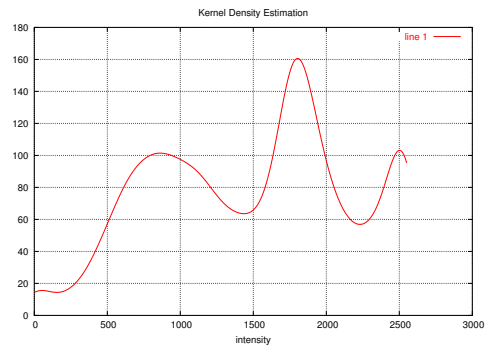
In Figure 3.2, since visualization is easier, we present the histogram (a) and kernel density estimation (b) of brightness of an image. Note that, the histogram is discrete and as seen in Figure 3.2b it produces a noisy estimate. However, kernel density (Figure 3.2c) produces a continuous and smooth estimate.



(a)



(b)



(c)

Figure 3.2: (a) Color image. (b) Histogram (250 bins) and (c) kernel density estimate of the intensity values (gray level image).

3.2 Texture

Texture is a measure of the variation of the intensity of a surface, quantifying properties such as smoothness, coarseness and regularity [HO85]. It's often used as a region descriptor in image analysis and computer vision. Compared to color, texture usually require a post

processing step to generate texture descriptors, which are generally illumination invariant. There are various approaches to measure the texture.

- *Gray level co-occurrence matrix* (GLCM) defines a two dimensional histogram that shows the occurrence of intensities in a specified direction and specified distance [HSD73, Lam96]. Since GLCM is based on raw color intensities, it changes as the illumination changes.
- *Law's texture measures* are computed by convolving the image with 25 two-dimensional filters generated from five one-dimensional filters corresponding to level, edge, spot, wave and ripple [Law80]. This approach requires very heavy computation due to convolution of the image with 25 two-dimensional filters.
- *Filter banks* have been the most popular texture measures during the last decade. Generally, a selection of filters (high-pass, low-pass or bandpass), which are composed of various scales and orientations, are applied to the image. Commonly used filters in this regard are different families of wavelets [Mal89], steerable pyramids [FA91] and bank of bandpass filters [JF91, BB94]. For a good reference on degree of discrimination of most filter based approaches suitable for infrared imagery, interested readers are directed to [YSS03, Bov91].

Due to better modeling performance (as outlined in [YSS03]), we use the steerable pyramids for texture representation. Steerable pyramids are multi-scale and multi-oriented linear basis, where the resulting filter coefficients are complex valued. The real and the imaginary

parts correspond to even and odd basis [FA91, GBG94, PS00]. Compared to using other filter banks, steerable pyramids perform a polar-separable decomposition in the frequency domain, thus allowing independent representation of scale and orientation. Using steerable pyramid filters, the image can be divided into its residual high-pass band (HP), multi-scale low-pass bands (LP_i) and multi scale sub-bands (BP_i) as depicted in Figure 3.3. The recursive steerable pyramid representation of an image is obtained by first convolving an image with a high-pass filter, following by a low-pass filter and finally convolving the low-pass response with a set of band-pass filters. This process of filtering continues by low-pass filtering subsampled response again iteratively, until a desired resolution is obtained.

Selecting the right filters for sub-band analysis plays a critical role in statistical modeling, which will be discussed in the next section. Usually, sub-bands are obtained using oriented filters, such as Gabor filters or Gaussian derivative filters. In order to have a disjoint feature space, such that the sub-bands are independent, we adopt the *Gabor wavelets*, which creates an orthonormal sub-band basis. Gabor wavelets are Gaussian modulated sine and cosine gratings:

$$G_i(x, y) = e^{-\pi[x^2/\alpha^2+y^2/\beta^2]} \cdot e^{-2\pi i[u_0x+v_0y]}$$

where α and β specify effective width and height, while u_0 and v_0 specify the modulation of the filter [BB94].

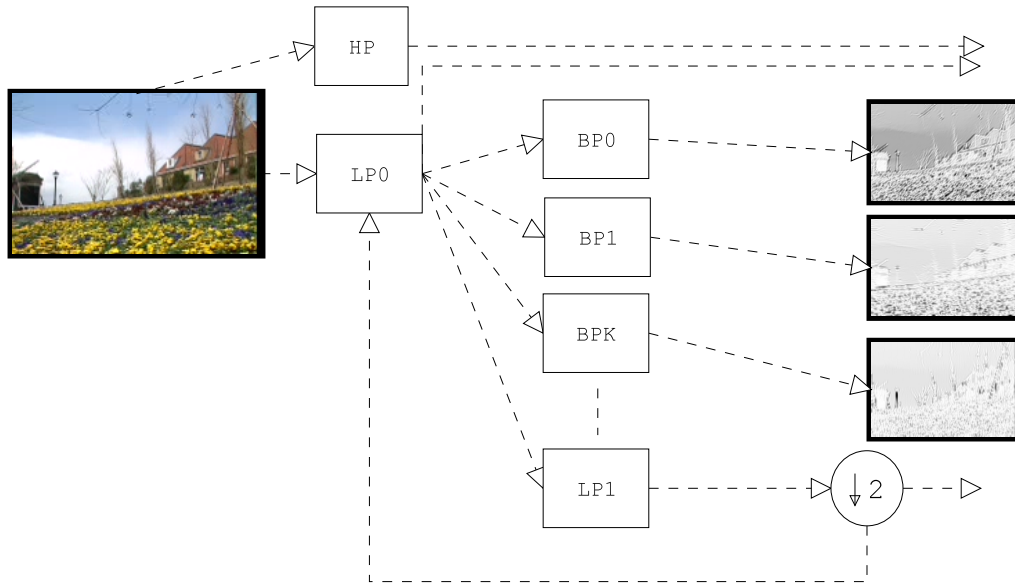


Figure 3.3: Steerable pyramid representation of an image. HP refers to the high-pass filter, LP0 and LP1 refer to low-pass filter, BP0, BP1,...BPK refer to band-pass filters. The images on the right are the magnitude responses of the band-pass filters for the input image given on the left.

3.2.1 Modeling Texture

A common texture representation is to create a multi dimensional feature vector obtained from various directional filters. This representation is then modeled using various models. For instance, Fleet *et al.* used the mixture model of phase response of Gabor filters in context of *tracking* [JFE01]; similarly Do and Vetterli used a single Gaussian model of wavelet decomposed textures for *texture retrieval* [DV02]; in the *texture recognition* field Leung used *k-means* clustering of magnitude response of various filter banks [Leu01]; Paragios and

Deriche used mixture model of magnitude of Gabor filter banks for *texture based segmentation* [PD02], similarly Hofmann *et al.* used local histograms of multi-scale Gabor filter banks [HPB98] for the same problem.

Similar to color, *PDFs* for orthonormal texture can be generated using kernel density estimation. However, in contrast to the color, since the size of the non-parametric model is not known, non-parametric density estimate is not efficient for modeling the texture. An alternative to non-parametric models is to use parametric models, such as mixture of Gaussians [JFE01, PD02]. In a mixture model, the probability of observing a value x is computed using model parameters (μ_k and σ_k for Gaussian) and *a priori* component probabilities, P_k , from:

$$p(x|\Theta) = \sum_{k=1}^{C_N} P_k p_k(x|\mu_k, \sigma_k), \quad (3.1)$$

where $\Theta = \{P_k, \mu_k, \sigma_k : k = 1 \dots C_N\}$. In Equation 3.1, the primary unknowns are the *a priori* component probabilities P_k , the mean μ and the standard deviation σ for each component and the secondary unknown is the number of components C_N . We select the number of components, C_N , manually, and the rest of the unknowns are computed using the *Expectation Maximization* (EM) algorithm outlined in [DH73] for the 2-dimensional space of complex filter responses. In Figure 3.4, a sample texture and its Gaussian mixture model in 2D are shown.

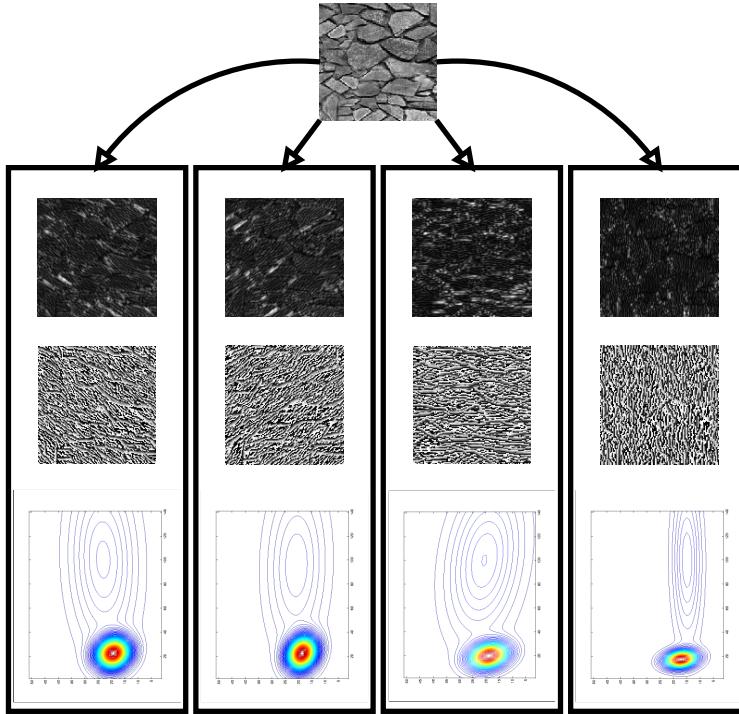


Figure 3.4: Sample texture (top image). Four boxes represent the four bandpass filter responses using Gabor wavelets in four directions. Inside each box, the first row is the magnitude of the response, second row is the phase response and last row is their mixture model computed using the EM algorithm for $C_N = 2$.

3.3 Local Intensity Deviation (LID) Feature

Local intensity deviation generates better object representations for very small objects (of size five to ten pixels) compared to the steerable pyramid, which smooths the image and loses the objects. The LID is computed for each pixel, \mathbf{x}_i , in its neighborhood defined by M

using:

$$S(\mathbf{x}) = \sqrt{\frac{1}{|M|-1} \sum_{\mathbf{x}_i \in M} (I(\mathbf{x}_i) - I(\mathbf{x}))^2}, \quad (3.2)$$

where $I : \mathbb{N}^2 \rightarrow \mathbb{N}^1$ is the imaging function. Although the size of the window affects the performance, from our experiments, we found out that fixing it to 5x5 results in better performance for small objects. In Figure 3.5, we show the standard deviation image generated using Equation 3.2. As can be seen, the object regions are clearly emphasized.

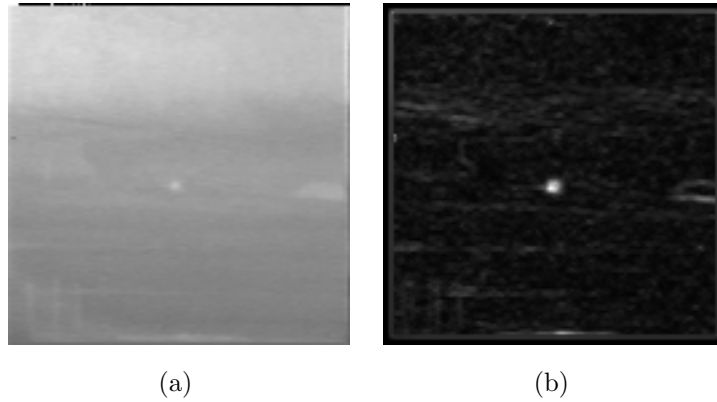


Figure 3.5: Local standard deviation feature: (a) input image, (b) resulting local standard deviation.

Following the descriptions for the color modeling given in Section 3.1.1, we model the LID using kernel density estimation, where the kernel is the Epanechnikov kernel.

3.4 Fusing Features

Fusing various features is usually performed to enhance the decision making process of method proposed for a vision problem (segmentation, tracking, recognition etc.) and it has long been considered by researchers in the field. In [Bla92, TAE98, KS01, HE02], the authors used motion, brightness and/or boundary cues for segmentation of layers in a video. In [JFE01], Fleet et al. used phase response of Gabor filter along with spatial pixel information for object tracking. Possible approaches to combine features are

- Cascaded integration,
- Supervised late integration,
- Unsupervised early integration.

Without loss of generality, we will discuss the integration methods on a simple *two-class* classification problem, such that an object can either be from class **A** or class **B**.

3.4.1 Cascaded Integration

Based on only one feature, classification of an object can easily be performed by the certainty of assigning the object to class **A** or class **B**. An intuitive extension to this, upon availability of an additional feature, is to increase or decrease the certainty computed by the first feature and reclassify the object, such that additional feature is cascaded to the first one. In some

cases with more features, the certainty can be increased. In [Bla92] and [TAE98], the authors followed this approach and considered each feature one at a time in different steps of the algorithm. For instance, Black [Bla92] first used intensity to classify pixels in the image, and refined the classification by introducing motion information based on coherent motion assumption for pixels that belong to the same region. In Figure 3.6, we show this coarse to fine approach. One important property is that the decision may change at every step according to the certainty of the feature.

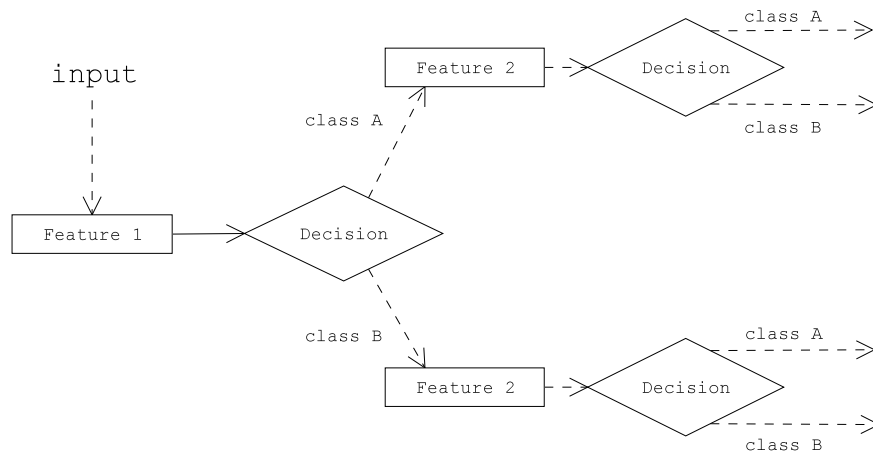


Figure 3.6: Cascaded integration decision flow.

The basic disadvantage of cascaded information is that the features are equally weighted and the decision can change at every step of the flow. In addition, using features one at a time results in a dependency of the current classification on previous steps, which may be erroneous.

3.4.2 Supervised Late Integration

For some classification problems a set of features might have more priority than the others. In such cases, we may want to weigh them more than the others, where the weights need to be defined manually. Due to the weighting mechanism, the final decision on classification is deferred until all the features are considered.

For instance, in the context of video segmentation, Khan and Shah [KS01] combined the features through manually assigned weights, which emphasized some features more than others in a *linear opinion polling* scheme. Their method uses a semi-automatic weight assignment depending on the observation that brightness feature is more stable along object boundaries. Similarly, Yilmaz *et al.* [YSL01] used convex combination of brightness and texture for tracking in infrared imagery, where the weights were manually defined.

The decision flow of the late integration approach (which is also called as *linear opinion polling*) is given in Figure 3.7. As observed, the decision for each feature is obtained independent of the others, and they are combined with manually defined weights, using:

$$P(\alpha|x) = \prod_{\beta} \left(\frac{P(x|\alpha)p(\alpha)}{p(x)} \right),$$

where $\alpha = \{\mathbf{A}, \mathbf{B}\}$ (the set of classes) and β is the set of features (texture, color, motion, etc.).

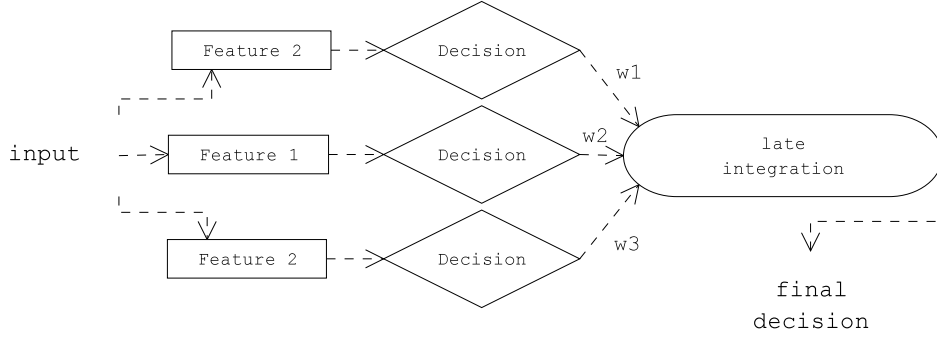


Figure 3.7: Supervised late integration decision flow.

3.4.3 Unsupervised Early Integration

Due to the manual weighting scheme in *linear opinion polling*, indiscriminative features might have equal or even higher weights, which may lead to false classification. This problem can be solved by unsupervised early integration (*independent opinion polling*). Independent opinion polling strategy relies on combining independent features according to their discrimination power and is evaluated prior to classification. To simplify the discussion, let there be two classes, namely the object and the background classes, and a pixel can be in either of the classes. The total likelihood of the observations for the object or the background model is given by

$$p(x|M_\alpha) = \prod_{\beta} p_{\beta}(x|M_{\alpha,\beta})$$

where $\alpha \in \{\text{object, background}\}$, $\beta \in \{\text{color, \{steerable sub-bands\}}\}$ and x is spatial variable (pixel coordinate). Using the Bayes' rule, a posteriori estimate of membership can be

computed using,

$$p(\alpha|x) = \frac{\prod_{\beta} p_{\beta}(x|M_{\alpha,\beta})p(\alpha)}{\sum_{\alpha} \prod_{\beta} p_{\beta}(x|M_{\alpha,\beta})p(\alpha)} \quad (3.3)$$

It can easily be observed from Equation 3.3 that features which are not discriminative will have equal membership probabilities; whereas the discriminant features will be emphasized. The decision flow of this scheme using the Equation 3.3 is presented in Figure 3.8. As can be observed all the features are combined together prior to any decision making and the output of this combination is used for the final decision making process.

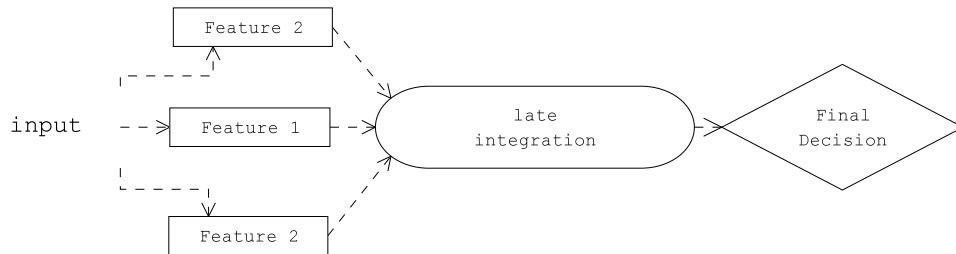


Figure 3.8: Unsupervised late integration decision flow.

Late unsupervised integration has been successfully used in the context of video segmentation in [HE02, TSA01]. In this thesis, we extend this scheme to the object tracking problem for fusing different visual cues.

3.5 Summary

There are various features that can be used for representing objects, such as color, depth, motion and texture. Since most of the computer vision problems are ill-posed, more features increase the robustness of solutions. In this chapter, we summarized the color, texture and local standard deviation features and discussed possible models used in the vision community. In addition, possible approaches on how to fuse subsets of these features is detailed. In the rest of the thesis, we will use these features to represent the objects for both of the proposed object tracking approaches.

CHAPTER 4

OBJECT TRACKING IN INFRARED IMAGERY CAPTURED FROM AIRBORNE VEHICLES

Detection and tracking of moving or stationary objects in forward looking infrared (FLIR) imagery are challenging research topics in computer vision. Though a great deal of effort has been expended on detecting and tracking objects in visual images, there has been only limited amount of work on thermal images in the computer vision community.

The thermal images are obtained by sensing the radiation in the infrared (IR) spectrum, which is either emitted or reflected by the object in the scene. Due to this property, IR images can provide information which is not available in visual images. However, in contrast to visual images, the images obtained from an IR sensor have extremely low signal to noise ratio (SNR), which results in limited information for performing detection or tracking tasks. In addition, in airborne FLIR images, non-repeatability of the target signature, competing background clutter, lack of *a priori* information, high ego-motion of the sensor and the artifacts due to the weather conditions make detecting or tracking objects even harder.

In this chapter, we present an approach for real-time object tracking in FLIR imagery in the presence of high global motion and changes in target features, i.e., shape and intensity. Moreover, the objects are not required to have constant velocity or acceleration. The proposed tracking algorithm uses the positions and the sizes of targets determined by the object detection scheme. For object detection, we apply steerable filters and compute texture energies of the objects, which are located using a segmentation based approach. Once the objects are detected, the tracking method employs three modules to perform tracking. The first module, which is motivated from [CRM00], is based on finding the translation vector in the image space that minimizes the distance between the distributions of the model and the candidate. The distributions are obtained from the intensity and local standard deviation measure of the frames. The local standard deviation measure is obtained in the neighborhood of each pixel in the frame and provides a very good representation of frequency content of the local image structure. Based on the distance measure computed from the object feature distributions, the other two modules compensate for the sensor ego-motion and update the object model. The global motion estimation module uses the multi-resolution scheme of [BAH92] assuming a planar scene under perspective projection. It uses Gabor filter responses of two consecutive frames to obtain the pseudo-perspective motion parameters.

4.1 Tracking Model

The probability density functions (PDF) of color or LID features as described in Section 3.1.1, do not contain spatial relationship of the intensities. To incorporate spatial information, kernel density estimation is defined by cascading two Epanechnikov kernels. The first kernel is used to define the spatial relationship of the feature using the Euclidean distance of its spatial position from the object centroid, i.e., we place a two-dimensional kernel centered on the object centroid and the kernel values as spatial weights. Two-dimensional Epanechnikov kernel is given by

$$K_2(\mathbf{x}) = \begin{cases} \frac{2}{\pi h^2}(h^2 - \mathbf{x}^T \mathbf{x}) & \mathbf{x}^T \mathbf{x} < h^2 \\ 0 & \text{otherwise} \end{cases},$$

where h is the radius of the kernel [CRM00]. The second kernel is used as a weighing factor in the feature histogram, i.e., we place a one-dimensional kernel centered on the feature value and use the kernel values as weights. One dimensional Epanechnikov kernel is given by

$$K_1(x) = \begin{cases} \frac{3}{4h}(h^2 - x^2) & x < h \\ 0 & \text{otherwise} \end{cases},$$

such that $\int_{-h}^h K_1(x)dx = 1$. Using the cascaded kernels, density estimate of feature u for an object can be estimated from

$$P(u) = \frac{\sum_{i=1}^n K_1(I(\mathbf{x}_i) - u) K_2(\mathbf{x}_i - \mathbf{m})}{C}, \quad (4.1)$$

where C is the normalization constant, $C = \sum_{i=1}^n K_2(\mathbf{m} - \mathbf{x}_i)$ and the bandwidths, h_1 and h_2 for both kernels are specified separately. Note that, the discussion presented here does not deal with fusing the different features as discussed in Section 3.4.

4.2 Methodology

Assume that the object first appeared in the 0^{th} frame, and \mathbf{m}_0 denotes its center. To track the object in the succeeding frames, kernel density estimates of each *bin* for both the intensity, Q_I , and the LID, Q_S , of images are computed using Equation 4.1.

Using the object model defined in Section 4.1, one possible way to find the object position in the current frame is to search neighboring regions for a distribution similar to the model computed for different scales of two-dimensional kernel in the neighboring regions [YS02a]. Although compared to template matching based methods, such an approach is more stable to changes in the object features such as shape, it is still computationally expensive. Rather, we will locate the object position directly by minimizing the distance between the model and the candidate *PDFs*, which is defined by

$$d(\mathbf{m}) = \sqrt{1 - \rho(\mathbf{m})}, \quad (4.2)$$

where $\rho(\mathbf{m})$ is the modified Bhattacharya coefficient which fuses the information obtained from the two different modalities: intensity and LID. Considering Equation 4.2, the modified Bhattacharya coefficient can also be interpreted as a likelihood measure between the model

and the candidate distributions, and is given by

$$\rho(\mathbf{m}) = \sum_{u=1}^b \left(\lambda \underbrace{\sqrt{P_{I_u}(\mathbf{m})Q_{I_u}}}_{\text{intensity based}} + (1 - \lambda) \underbrace{\sqrt{P_{S_u}(\mathbf{m})Q_{S_u}}}_{\text{stdv. based}} \right),$$

where b is the number of bins common to both the intensity and LID distributions and $\lambda \in [0, 1]$ is the parameter which balances the contribution of intensity and LID features.

Expanding $\rho(\mathbf{m})$, the likelihood, using Taylor series around previous object position \mathbf{m}_0 gives:

$$\begin{aligned} \rho(\mathbf{m}) = & \rho(\mathbf{m}_0) + \frac{1}{4C} \sum_{i=1}^n K_2(\mathbf{m} - \mathbf{x}_i) \left(\sum_{u=1}^b K_1(I(\mathbf{x}_i) - u) \sqrt{\frac{Q_{I_u}}{P_{I_u}(\mathbf{m}_0)}} + \right. \\ & \left. \sum_{u=1}^b K_1(S(\mathbf{x}_i) - u) \sqrt{\frac{Q_{S_u}}{P_{S_u}(\mathbf{m}_0)}} \right), \end{aligned} \quad (4.3)$$

where $S(\mathbf{x}_i)$ denotes the normalized LID measure computed using Equation 3.2. Discarding the constant terms in Equation 4.3, the likelihood of each pixel belonging to the object can be defined by the following function:

$$\begin{aligned} \psi_i = & \sum_{i=1}^n K_2(\mathbf{m} - \mathbf{x}_i) \left(\sum_{u=1}^b K_1(I(\mathbf{x}_i) - u) \sqrt{\frac{Q_{I_u}}{P_{I_u}(\mathbf{m}_0)}} + \right. \\ & \left. \sum_{u=1}^b K_1(S(\mathbf{x}_i) - u) \sqrt{\frac{Q_{S_u}}{P_{S_u}(\mathbf{m}_0)}} \right), \end{aligned} \quad (4.4)$$

where $i = 1 \dots n$. In other words, ψ_i denotes the likelihood of the object in the spatial kernel defined by K_2 . Here, we assume ψ is normalized such that $\sum_{i=1}^n \psi_i = 1$. Starting from the center of a differentiable kernel like K_2 , a hill-climbing scheme can be used to maximize the likelihood of the object. During the maximization, density estimate given by

$$f_{\kappa}(\mathbf{m}) = \frac{1}{nh^d} \sum_{i=1}^n K(\mathbf{x}_i - \mathbf{m}), \quad (4.5)$$

will have a gradient direction, $\hat{\nabla}f$, pointing to the new kernel center at every iteration [Che95, FH75]. In Equation 4.5, \mathbf{m} is the center of a d -dimensional kernel, and n is the number of points inside the kernel. Translating the image origin to the kernel center, such that $\mathbf{m} = 0$, the mean shift vector is given by

$$\mathbf{m}_1 = \mathbf{m}_0 + \frac{4}{\pi h^4} \sum_{i=1}^n \psi_i \mathbf{x}_i, \quad (4.6)$$

where \mathbf{m}_1 is the new object position.

The scheme outlined above can be used for tracking objects whose features remain constant throughout the sequence. However, in general, objects don't exhibit constant brightness or contrast, thus the object feature distributions may change. The following section presents a solution to overcome this shortcoming.

4.3 Object Model Update

During the course of tracking, object features may abruptly change. The simplest way to change the object model is to periodically update the feature distributions. However, due to the low contrast of an object with its background, which usually occurs in FLIR imagery, the update may not necessarily occur when the object is correctly localized. Another straightforward solution is to change the object model using a constant threshold on the similarity metric used in tracking. For instance, for correlation based methods, the model can be updated if the correlation of the model with the object is higher than a threshold. Similarly,

for the method outlined in Section 4.2, the model can be updated if the distance calculated in Equation 4.2 is lower than a threshold [YSL01]. The problem with this is to select the right threshold for all the sequences, i.e., a particular threshold may work very well for one sequence, but may fail for others.

The model can be automatically updated using sequence-specific information about the rate of change of object features, which can be computed using the distance measure given in Equation 4.2. In our implementation, the object model consists the object intensity and the LID distributions. The rate of change for object intensity and LID differs from one sequence to another. To utilize sequence-specific changes of the object features over time, the distribution of the distance (see Figure 4.1 for an example distribution) is modeled by a Gaussian distribution. The Gaussian distribution parameters, mean μ and standard deviation σ , are updated at each frame using:

$$\mu_k = \frac{(k-1)\mu_{k-1} + d_k}{k}, \quad (4.7)$$

$$\sigma_k^2 = \sigma_{k-1}^2 + (\mu_k - \mu_{k-1})^2 + \frac{d_k - \mu_k}{k-1}, \quad (4.8)$$

where k denotes the current frame number. The decision whether to update the model is made based on the current value of the distance d_k , i.e., if $d_k < m_k - 2\sigma_k$, then object model is updated. This scheme guarantees that the model is updated if the object distribution change is within the acceptable range for a particular sequence. For instance, for a sequence where the object distribution changes very rapidly, such that the μ_k and σ_k are high, the acceptable range for model update will be wide.

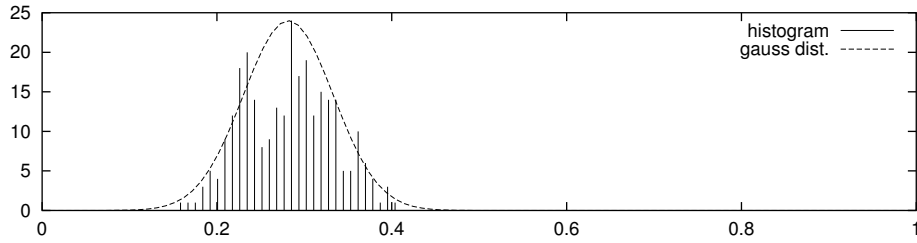
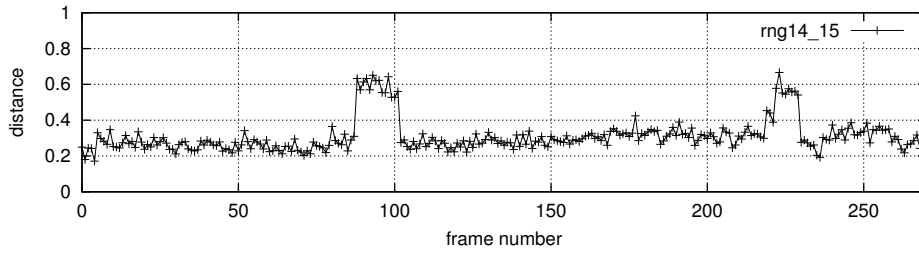


Figure 4.1: The histogram of distances calculated using Equation 4.2, and Gaussian model fitted to the distribution.

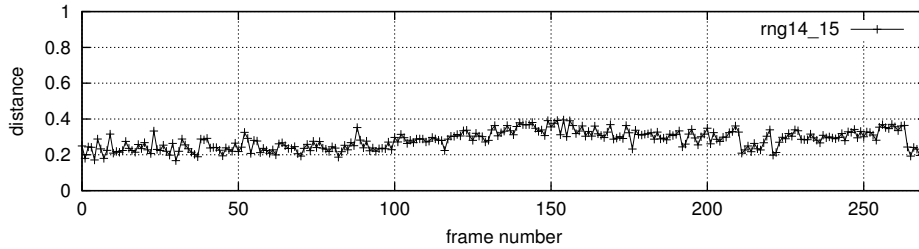
In Figure 4.2, distances between the model and the object distributions with and without model update are shown. In Figure 4.2a, the big jumps in the distance plot are due to the intensity change of the object for some parts of the sequence. These problems are corrected by model updating as shown in Figure 4.2b. More results are given in Section 4.6

4.4 Ego-Motion Compensation

The main limitation of the mean-shift based tracker is that at least some part of the object in the next frame should reside inside the kernel [CRM00]. FLIR sequences obtained via an airborne moving platform suffer from abrupt discontinuities in motion due to sensor ego-motion. Because of this, the output of the tracker becomes unreliable, and requires compensation of the sensor ego-motion.



(a)

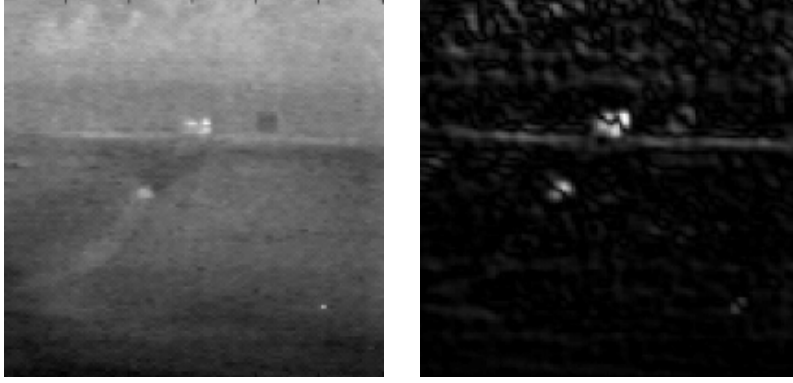


(b)

Figure 4.2: Distribution of the distances computed (a) before model update, (b) after model update.

There are several approaches presented in the literature for ego-motion (global) compensation. However, they are not directly applicable to FLIR imagery due to lack of texture and low SNR. For instance, we have noted that motion compensation using the intensity values does not result in good estimation of motion parameters.

Therefore, we employ images filtered by 2D Gabor filter kernels, which is given in Section 3.2. In our implementation, we used the real parts of the Gabor responses for four different orientations. The responses of the Gabor filters are then summed and used as the input for the global motion compensation module. In Figure 4.3, we show a selected frame from one of the infrared sequences along with its Gabor response.



(a)

(b)

Figure 4.3: (a) Sample frame from one of the infrared sequences, (b) summation of four Gabor responses of the frame in (a).

To compensate for the global motion, we employ the multi-resolution framework of [IA98b]. The compensation method uses the pseudo-perspective motion model given by:

$$\begin{aligned}
 u &= a_1 + a_2x + a_3y + a_4xy + a_5x^2, \\
 v &= a_6 + a_7x + a_8y + a_4y^2 + a_5xy,
 \end{aligned}$$

where $a_1 \dots a_8$ are motion parameters, (x, y) is the position of a pixel in image plane and (u, v) is the optical flow vector. Pseudo-perspective motion model provides a better estimate of the motion for the planar scenes in closing FLIR sequences compared to simpler motion models such as the affine, which fails to detect skew, pan and tilt in planar scenes. Rewriting the pseudo perspective motion of the sensor between two images in the matrix form we have:

$$\mathbf{U} = \mathbf{M}\mathbf{a}, \tag{4.9}$$

where $\mathbf{U} = (u, v)^T$ is the optical flow, $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)^T$ and

$$\mathbf{M} = \begin{pmatrix} 1 & x & y & xy & x^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & y^2 & xy & 1 & x & y \end{pmatrix}.$$

Optical flow can be computed using the optical flow constraint equation given by

$$\mathbf{F}_{\mathbf{x}}^T \mathbf{U} = -f_t, \quad (4.10)$$

where $\mathbf{F}_{\mathbf{x}} = (f_x \ f_y)^T$ is the spatial gradient vector and f_t is the temporal derivative.

Combining Equations 4.9 and 4.10 results in a linear system that can be solved using the least squares method as

$$\left(\sum \mathbf{M}^T \mathbf{F}_{\mathbf{x}} \mathbf{F}_{\mathbf{x}}^T \mathbf{M} \right) \mathbf{a} = - \sum f_t \mathbf{M}^T \mathbf{F}_{\mathbf{x}}. \quad (4.11)$$

In Figure 4.4, we show two successive frames I_k and I_{k+1} , their difference $I_{k+1} - I_k$, the compensated global motion, i.e., first frame registered onto the second frame, I'_k using Equation 4.11, and the difference $I'_k - I_{k+1}$. As it is clearly seen, the frames are correctly registered.

Compensating for sensor ego motion in images lacking adequate gradient information suffers from the biased estimation of the motion parameters. Especially for FLIR imagery, background clutter and lack of texture increase the possibility of estimating incorrect parameters. Based on these observations, it is important not to perform motion compensation for every frame. Similar to the scheme described in Section 4.3, we compensate for the global motion if the distance (computed using Equation 4.2) $d_k > m_k + 2\sigma_k$, where m_k is the mean

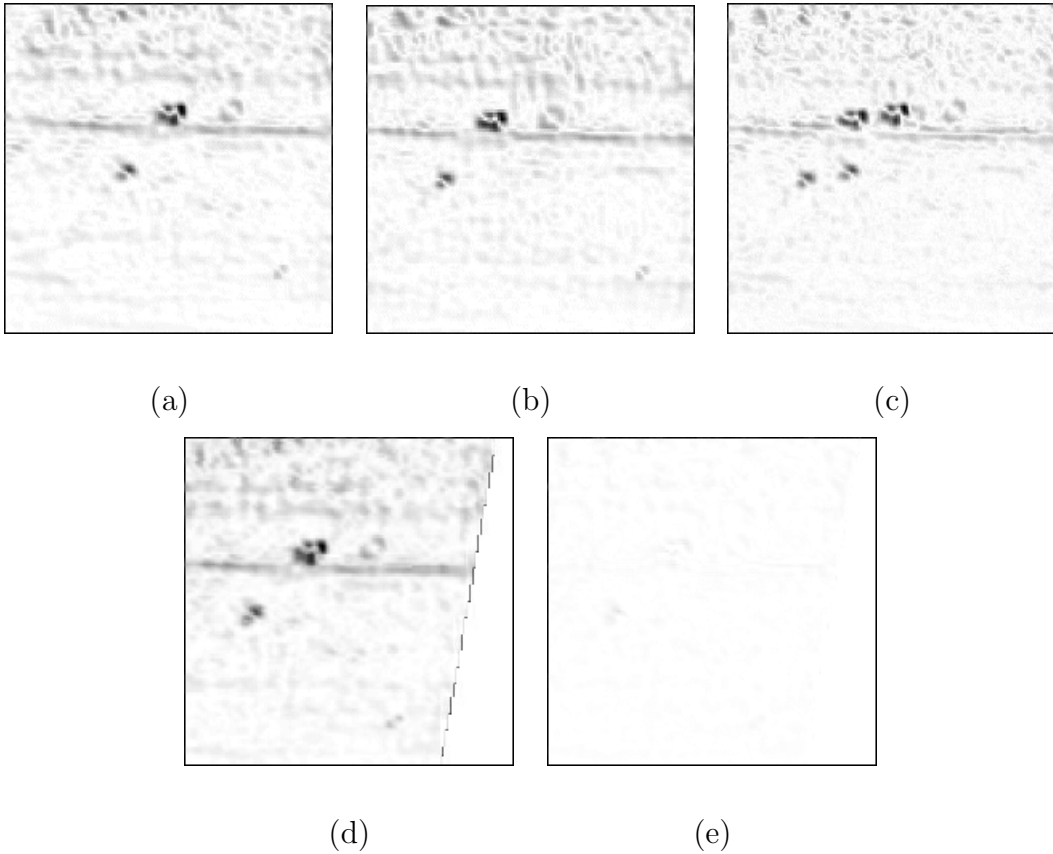


Figure 4.4: (a) The reference frame I_k , (b) the current frame I_{k+1} , (c) the difference image obtained using (a) and (b). Note the large bright spots due to miss registration. (d) First frame registered onto the second frame, i.e., global motion is compensated, (e) the difference image obtained using (b) and (d).

and σ_k is the standard deviation of d_i for $i < k$. This scheme guarantees compensation of global motion if the object distribution changed drastically for a particular sequence, such that the tracker fails to locate the object.

Once the projection parameters, \mathbf{a} , are computed by solving the system of equations given in Equation 4.9, we apply the transformation to the previous object center and compute the approximate new candidate object center using $\mathbf{m}_k = \mathbf{m}_{k-1} + \mathbf{U}$.

We then perform the mean shift computations in the neighborhood of the new object position to increase the likelihood of the object model and the new candidate model.

4.5 Algorithm

In our algorithm, we assume that the object first appeared in frame 0, and the current frame is k . The model distributions for intensity and LID are denoted by Q_I and Q_S respectively. Similarly, we denote the candidate distributions for intensity and LID for frame $k + 1$ by P_I and P_S respectively. The distance between the model and candidate distributions is referred to as d_k .

For the detected objects (by one of the methods described in Section 2.3), the algorithm executes the following steps to perform tracking in frame k . In our framework, we used the “Feature Clustering” approach for target detection, which is discussed in Section 2.3.3.

1. For the detected objects compute Q_I and Q_S using equation 4.1.
2. Initialize object center in frame k using the previous object center and compute distributions P_I and P_S .
3. Iteratively compute the modified mean shift vector using Equations 4.4 and 4.6.

4. Compute distance d_k (Equation 4.2), and go to step 2 until the change of d_k in each iteration is close to 0.
5. If $d_k < m_{k-1} - 2\sigma_{k-1}$ update Q_I and Q_S else if $d_k > m_{k-1} + 2\sigma_{k-1}$ compensate for the global motion.
6. Update the distance distribution parameters m_k and σ_k using Equations 4.7 and 4.8 then return to step 2.

Since each detected object in the scene has its own distribution model, the above tracking algorithm also performs tracking under partial occlusion. We do not address complete occlusions.

4.6 Experiments

We have applied the proposed tracking method to the AMCOM FLIR data set, which was made available to us in gray-scale format and has 41 sequences where each frame in each sequence is 128x128. The AMCOM data set is a very challenging data set. The original mean shift method fails to track the targets, due to presence of high sensor ego motion and noise in intensity feature, for most of the sequences provided in the dataset.

The proposed approach was developed in C++ on a Pentium III platform and the current implementation of the algorithm is capable of tracking at 10 fps. We used object detection methods where applicable, in other cases (for instance, dark objects), we manually marked

the object and performed tracking. We used 64 bins to construct distributions of the intensity and the LID feature PDFs.

In this section, we show the robustness of our approach using both the model update and motion compensation modules and present results on sequences which have low SNR and high global motion. In the results, rather than showing every 10th or 15th frame, we selected representative frames from these sequences to demonstrate motion of the objects. The positions of the objects in the sequences presented here are visually verified. For complete video sequences of results please visit Mean Shift www.cs.ucf.edu/vision/projects/MeanShift/MeanShift.html.

4.6.1 Object Tracking for Mobile Camera

In Figures 4.5, 4.6 and 4.7, we present the tracking results for sequences rng17_01, rng14_15 and rng19_07 respectively. The tracked objects in the images are marked by circles. In all three sequences, despite the fact that the objects look very similar to their backgrounds, their positions are correctly tracked.

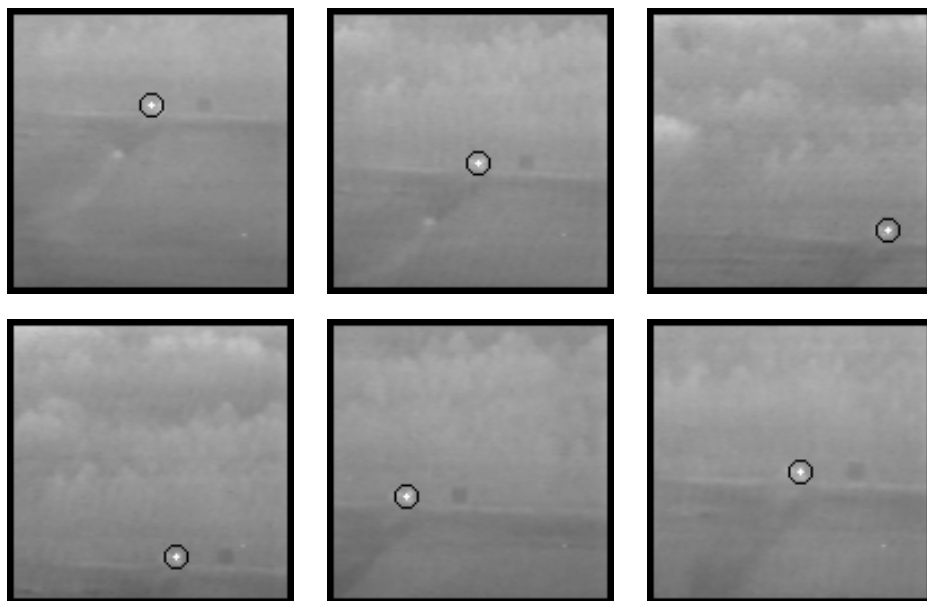


Figure 4.5: Tracking results for sequence rng17-01, frames 1, 17, 35, 53, 70 and 115.

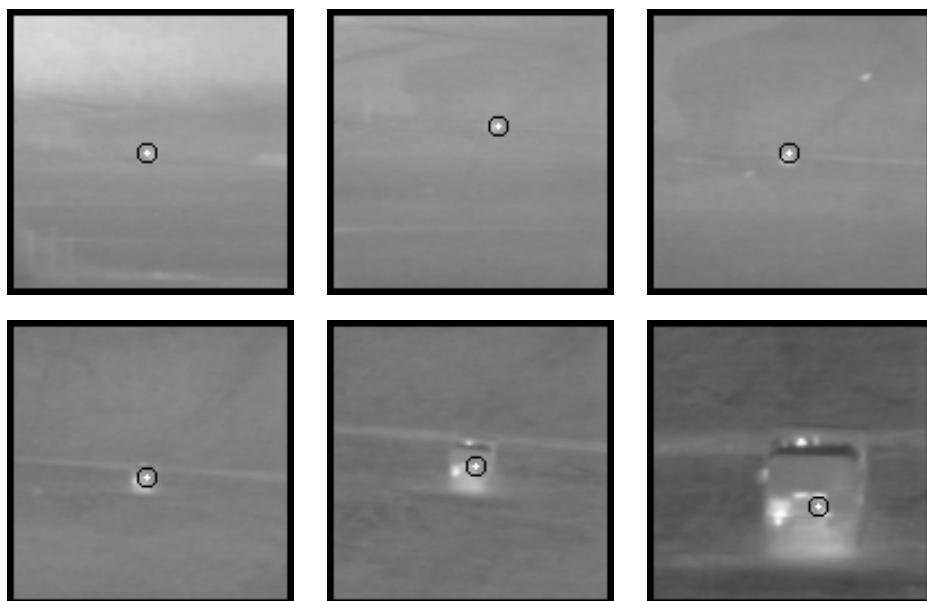


Figure 4.6: Tracking results for sequence rng14-15, frames 1, 60, 128, 195, 237 and 271.

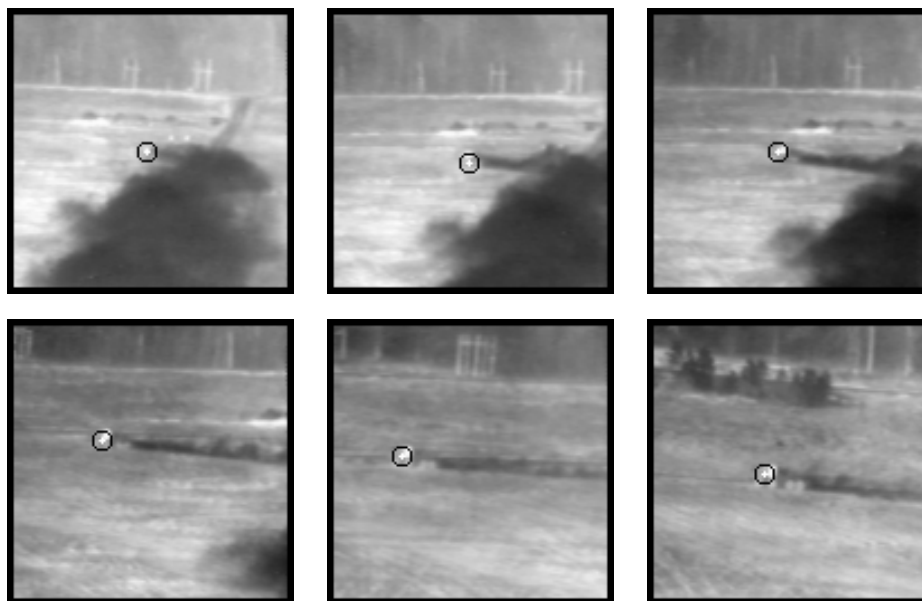


Figure 4.7: Tracking results for sequence rng19_07, frames 129, 138, 144, 159, 174 and 189.

4.6.2 Tracking Under High Sensor Ego-Motion

In Figure 4.8, we present the results that demonstrate the importance of using global motion compensation and automatic model update for two sequences. In the figure, first columns of both part (a) and part (b) show the tracking results where we neither update the object model nor compensate for global motion. The correct object positions are marked by 'x', whereas the detected object positions are marked by circles. Specifically in Figure 4.8a, the sensor ego-motion causes an abrupt change in the object position and the object is lost without global motion compensation; however as can be seen from the second column, the object is correctly located using the global motion compensation module. In Figure 4.8b, due to changes in object distribution over time and low SNR of the object, (as it is clear from the first column) the tracker loses the object when it does not perform the model update.

However, as shown in the second column the model update improves the performance and the object is correctly tracked.

4.6.3 Tracking Cold Infrared Objects

In Figure 4.9, tracking results for cold objects are shown. Since the detection method only detects *hot* objects, the sizes and positions of the objects are manually initialized. In both sequences, rng18_07 (part (a)) and rng18_03 (part (b)), the objects have very low contrast with the background and particularly in part (b) neighboring locations hide the object due to high gradient magnitude. Using both the intensity and the LID measures together improved the tracking performance for both sequences. In particular, for sequence rng18_07 the algorithm was able to track the object successfully even in presence of very high variation in the intensity levels.

4.6.4 Tracking Multiple Objects

To demonstrate the multiple object tracking capability, we also applied the tracking method to sequences with multiple objects. In Figures 4.10, 4.11 and 4.12, tracking results are

presented for sequences rng16_07, rng19_NS and rng16_18. As seen from the results, the objects are correctly tracked. Note that we do not address the complete occlusion problem.

4.7 Conclusions

We propose a robust approach for tracking targets in airborne FLIR imagery. The tracking method requires the position and the size of the target in the first frame. The target detection scheme, which is used to initialize the tracking algorithm, finds target candidates using fuzzy clustering, edge fusion and texture measures. We employ a texture analysis on these candidates to select the correct targets. The experimental results for 200 target and 200 non-target regions that were manually marked show that “steerable filters” have better categorization performance compared to “Law’s texture measures” and “wavelet filters”. Once the targets are detected, the tracking system tracks the targets by finding the translation of the target center in the image space using the intensity and local standard deviation distributions. According to the distribution of the distance calculated from target model and distributions of the new target center, the algorithm decides whether a model update or global motion compensation is necessary. Sensor ego-motion is compensated for two consecutive frames assuming the pseudo-perspective motion model. The results demonstrated on

sequences, which have low SNR and high ego motion, show the robustness of the proposed approach for tracking FLIR targets.

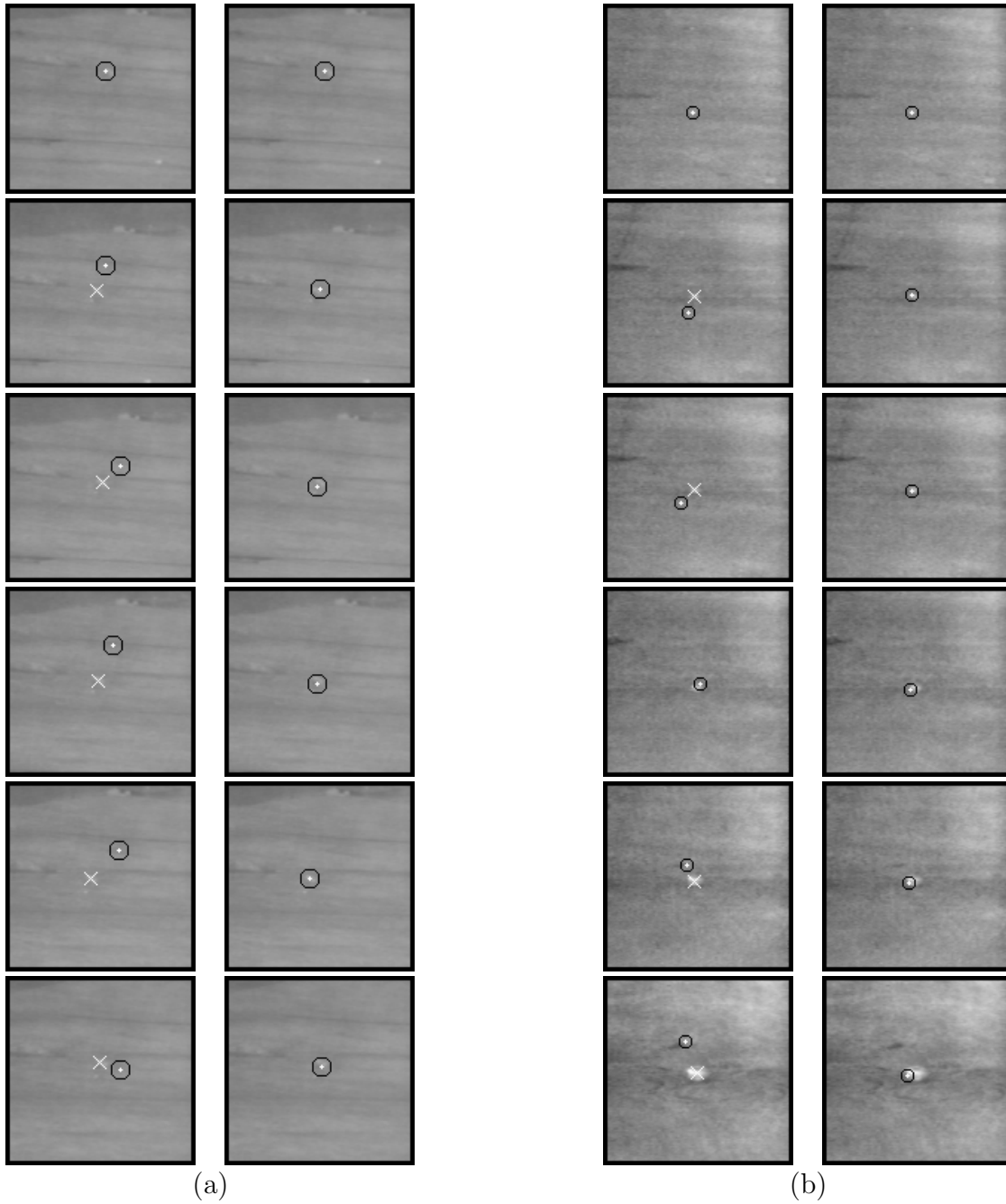
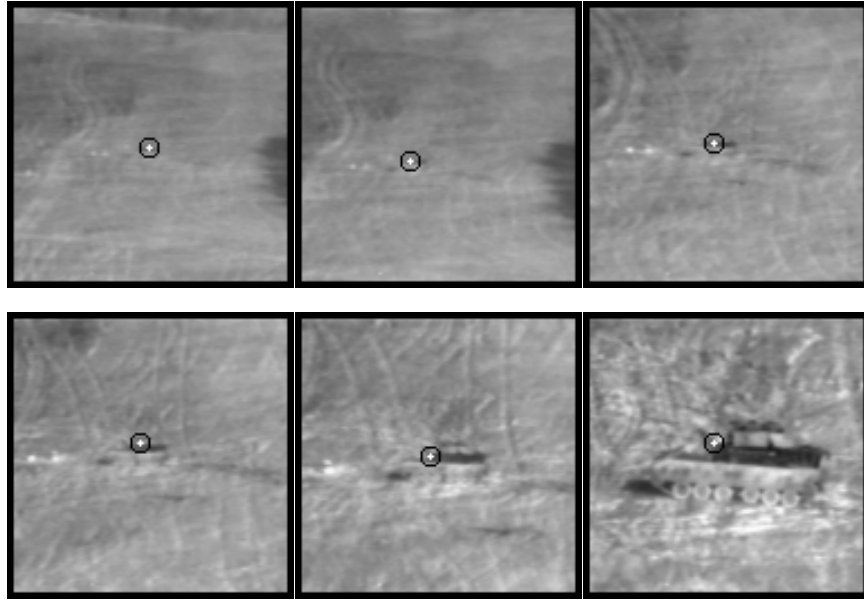
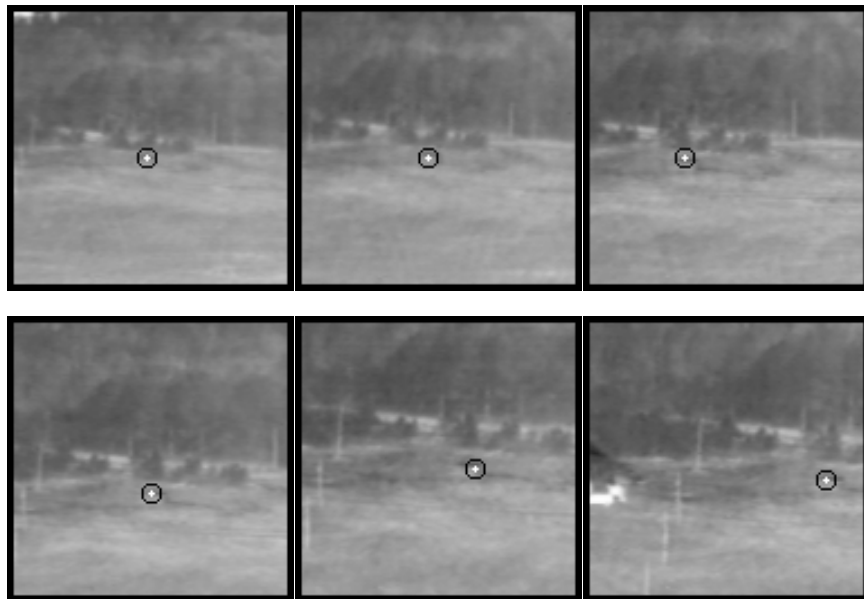


Figure 4.8: The first and the second columns show results with and without global motion compensation and model update respectively. (a) Sequence rng15_20, (b) sequence rng22_08. The correct positions are marked by \times , and the detected positions are marked by \bigcirc .



(a)



(b)

Figure 4.9: Object tracking results for cold objects: (a) sequence rng18_07, frames 113, 135, 181, 204, 229 and 259, (b) sequence rng18_03, frames 11, 39, 63, 91, 119, 154 and 170.

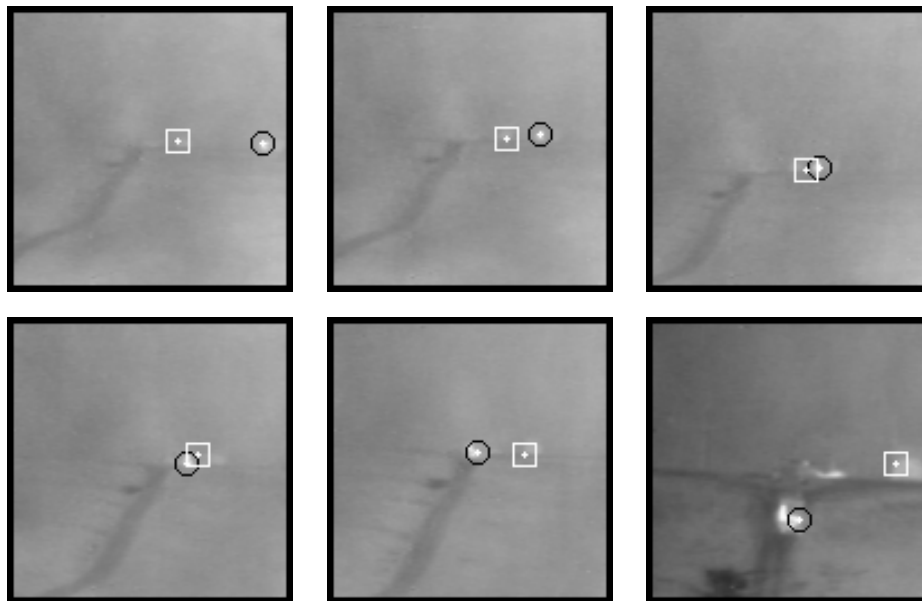


Figure 4.10: Tracking results for multiple objects in sequence rng16_07, frames 226, 241, 253, 274, 294 and 386.

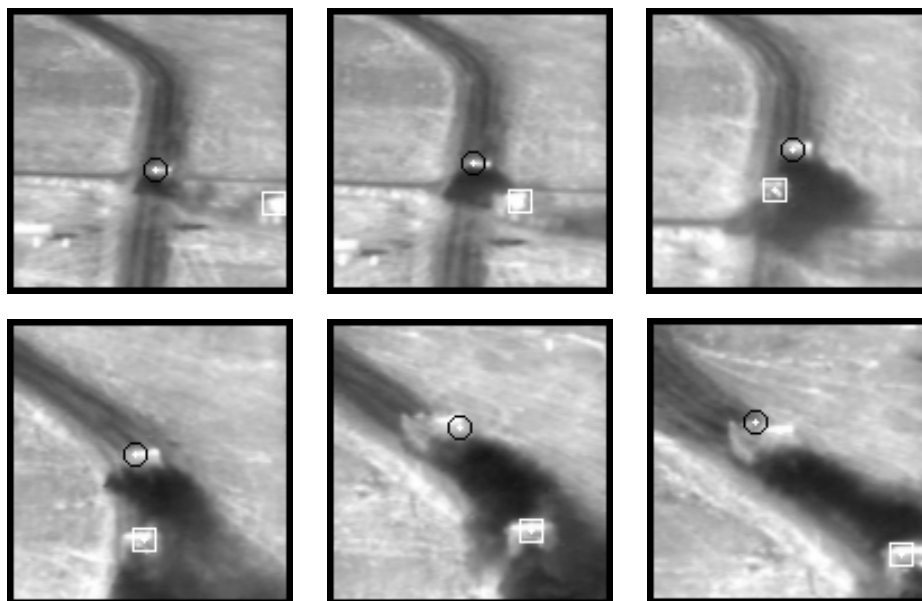


Figure 4.11: Tracking results for multiple objects in sequence rng19_NS, frames 208, 215, 231, 253, 267 and 274.

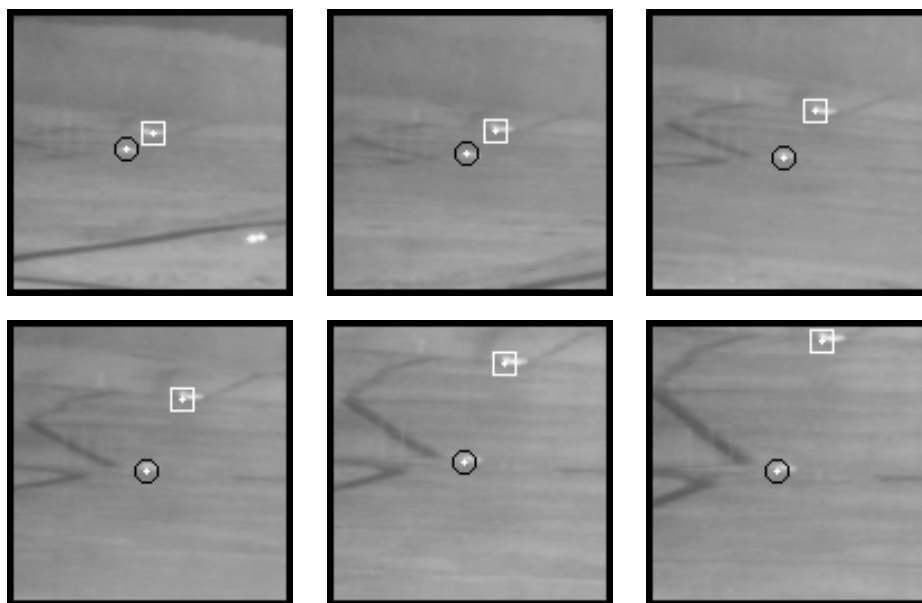


Figure 4.12: Tracking results for multiple objects in sequence rng16_18, frames 1, 20, 40, 60, 79 and 99.

CHAPTER 5

CONTOUR BASED OBJECT TRACKING WITH OCCLUSION HANDLING

In the previous chapter, we proposed an object tracker which used primitive geometric shapes, and was able to track the object's centroid. However, tracking objects centroid or an object region defined by simple geometric shapes is not adequate for high level vision tasks, such as fine level action recognition and behavior analysis, where detailed modeling of the shape deformation during an action is required. We believe, complete object contours (see Figure 2.1) are the best two-dimensional representations for performing high-level vision tasks. In this chapter, we propose a contour based object tracking method (see Section 2.4.3 for details on contour tracking), which tracks the complete object contours in video acquired using mobile cameras.

The flow diagram of the proposed method is given in Figure 5.1. The proposed method can track multiple objects, handle occlusions, and adapt to changing object and background appearances by applying on line models. We model the color and texture of the objects and their backgrounds by using a semi-parametric models, where the mixing parameters are

computed based on the strength of the discrimination of each feature. To deal with ambiguities caused by object occlusions, which result in missing observations in the estimated object regions, we use on line shape priors. Based on the appearance models, we perform discriminant analysis localized around the contour and rigorously derive a new energy functional that maximizes the likelihood of observing the object contour. Associated Euler-Lagrange equations, which are used to minimize the energy in the gradient descent direction, result in a system of nonlinear partial integro-differential equations of order one. To provide numerical stability during the minimization process, the contour is implicitly represented by level sets. The robustness of the proposed method is demonstrated on real sequences with and without complete object occlusions.

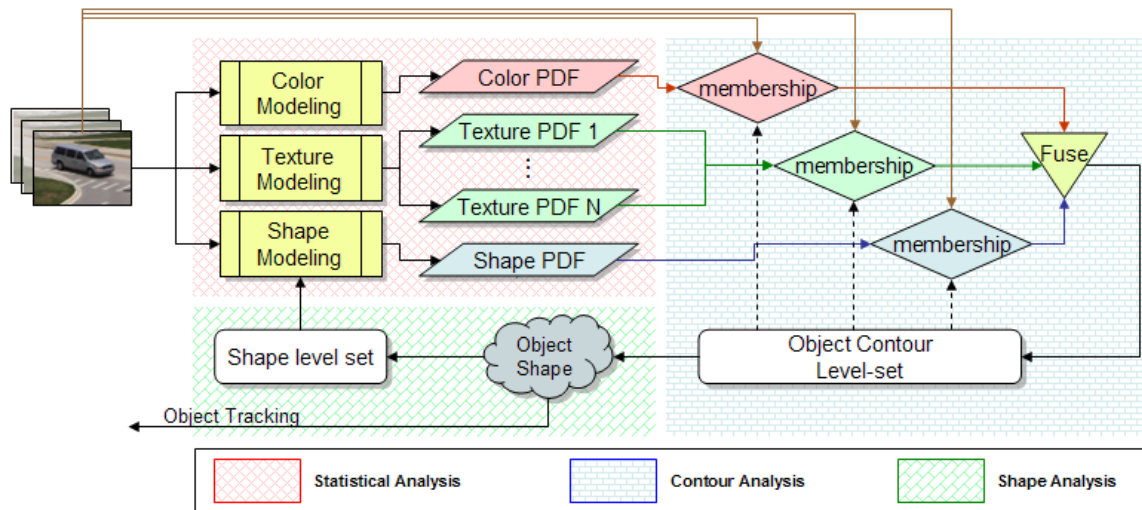


Figure 5.1: Flow chart of proposed contour tracking method.

5.1 Bayesian Framework For Object Tracking

Tracking an object in an image sequence $I^n : 0 < n < \infty$, can be treated as discriminant analysis of pixels on two non-overlapping classes, where the classes correspond to the *object region*, R_{obj} , and the *background region*, R_{bck} .

Without loss of generality, we can assume that in the spatial domain an image, I , is composed of an object region, R_{obj} (black region in Figure 5.2), and a background region, R_{bck} (light gray region in Figure 5.2), such that $I = R = R_{\text{obj}} \cup R_{\text{bck}}$. The boundary (*contour* shown with solid line between object and background regions in Figure 5.2), Γ , between these two regions is defined by the intersection of the directional curves corresponding to each region: $\Gamma = \Gamma_{\text{obj}} \cap \Gamma_{\text{bck}}$, where Γ_{obj} is border of object region (curve in counterclockwise direction of object region in Figure 5.2) and Γ_{bck} is the border of background region (curve in clockwise direction of object region in Figure 5.2).

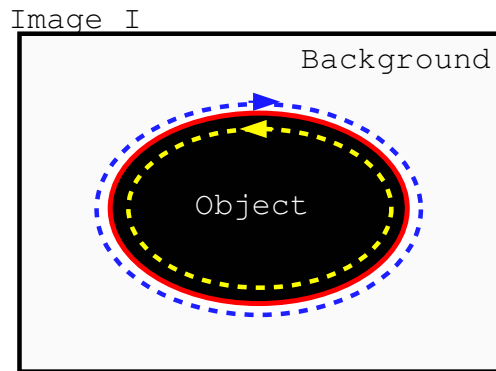


Figure 5.2: Definition of object and background regions in an image.

Assuming that there exists an object in the image, the likelihood of observing the boundary between the object and the background regions is equal to the likelihood of partitioning the image into two regions:

$$P(\Gamma) = P(\varphi(R) = \{R_{\text{obj}}, R_{\text{bck}}\}), \quad (5.1)$$

where φ is the partitioning operator [ZY96]. The posteriori boundary probability of Equation 5.1 can be used interchangeably with the posteriori probability of partitioning the space. Thus, for frame n , we formulate the object tracking problem in terms of the boundary probability, P_{Γ} , defined by the probabilities of the regions, given the current image, I^n , and the previous collection of object boundaries, $\Gamma^i : i = 0 \dots n - 1$, as

$$P_{\Gamma} = P(\varphi(R^n) | I^n, \Gamma^{n-1}, \Gamma^{n-2}, \dots, \Gamma^0). \quad (5.2)$$

where Γ^0 is the initial contour of the object.

The difference between the segmentation and the tracking is the availability of the object in the previous sequences, which is realized in Equation 5.2 by conditioning the current contour on previous contour observations. Using Bayes' rule,

$$P_{\Gamma} = \frac{P(I^n | \varphi(R^n), \Gamma^{n-1}, \dots, \Gamma^0) P(\varphi(R^n) | \Gamma^{n-1}, \dots, \Gamma^0)}{P(I^n | \Gamma^{n-1}, \dots, \Gamma^0)}. \quad (5.3)$$

Note that the denominator term in Equation 5.3 represents the probability of observing the intensity values in I^n given the previous contour observations, and is assumed to be constant, since the intensities conditioned on the shape of the object do not vary. Thus in the remainder of the derivation, we will denote

$$C = P(I^n | \Gamma^{n-1}, \Gamma^{n-2}, \dots, \Gamma^0).$$

The partitioning operator $\varphi(R^n)$ results in two separate partitions namely R_{obj} and R_{bck} .

Because of this, $P(I^n|\varphi(R^n), \Gamma^{n-1}, \dots, \Gamma^0)$ splits into two:

$$P(I^n|R_{\text{obj}}^n, \Gamma^{n-1}, \dots, \Gamma^0) \text{ and } P(I^n|R_{\text{bck}}^n, \Gamma^{n-1}, \dots, \Gamma^0).$$

Thus the probability of observing the contour becomes:

$$P_{\Gamma} = \frac{\overbrace{P(I^n|R_{\text{obj}}^n, \Gamma^{n-1}, \dots, \Gamma^0)}^{\text{object term}} \overbrace{P(I^n|R_{\text{bck}}^n, \Gamma^{n-1}, \dots, \Gamma^0)}^{\text{background term}} \overbrace{P(\varphi(R^n)|\Gamma^{n-1}, \dots, \Gamma^0)}^{\text{shape term}}}{C}. \quad (5.4)$$

The first two terms in Equation 5.4 reflects the probability of observing the color of a pixel, given the object and background regions along with the deformation of the contour overtime.

Once the models are generated for the object and the background regions, which takes the past contours into account, these terms can be computed using discriminant analysis. On the other hand, the last term is the partitioning probability purely based on the shape of the object observed over time and is not dependent on visual features (color, texture etc.).

This term can be computed by modeling observed shapes of the object overtime. To simplify the notion, without loss of generality, we will use $P_{R_{\alpha}}(I^n) = P(I^n|R_{\alpha}^n, \Gamma^{n-1}, \dots, \Gamma^0) : \alpha \in \{\text{obj}, \text{bck}\}$ and $P_S^n = P(\varphi(R^n)|\Gamma^{n-1}, \dots, \Gamma^0)$. Thus,

$$P_{\Gamma} = \frac{P_{R_{\text{obj}}}(I^n)P_{R_{\text{bck}}}(I^n)P_S^n}{C}.$$

Holes (Figure 5.3b) or noise (Figure 5.3c) inside the object is problematic for tracking methods (see discussion in Section 2.4). For instance, object detection methods, which are required for point trackers, may split the region into many sub-regions and tracking will

become erroneous. Similarly for trackers which estimate the transformation of the object represented by simple geometric shapes, will have poor motion estimation.

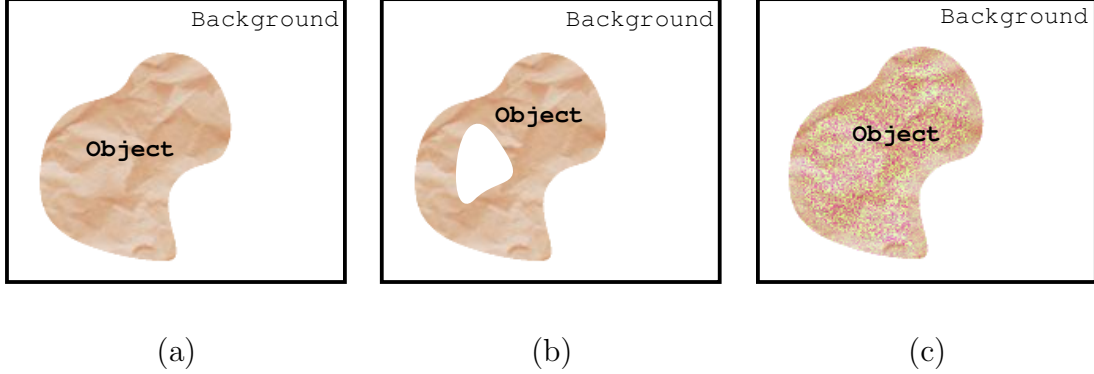


Figure 5.3: (a) An object region at time t ; (b) example of a hole or (c) noise appearing at time $t + 1$.

To address the problems outlined above, let there be two subregions, one inside the object region, $R_{\text{obj}}^{\Gamma} \subset R_{\text{obj}}$, and the other inside the background region, $R_{\text{bck}}^{\Gamma} \subset R_{\text{bck}}$, defined in the neighborhood of the contour. In Figure 5.4, we show the defined subregions for better visualization of the concept.

Limiting the region definition from the complete regions to the subregions shown in Figure 5.4 results in the modified contour probability, Γ' :

$$P_{\Gamma}' = \frac{P_{R_{\text{obj}}^{\Gamma}}(I^n)P_{R_{\text{bck}}^{\Gamma}}(I^n)P_S^n}{C}. \quad (5.5)$$

Note that $P_{\Gamma} \leq P_{\Gamma}'$ due to the discussion presented above. In the remainder of the paper, we replace the contour probability P_{Γ} with P_{Γ}' in Equation 5.5, and assume that each pixel

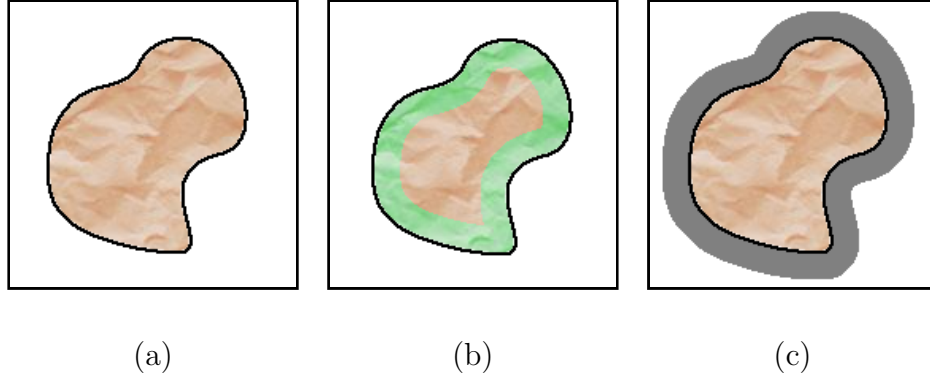


Figure 5.4: (a) The object region. (b) Selected band inside the object region, R_{obj}^Γ defined by the contour. (c) Selected band outside the object region, R_{bck}^Γ .

in the image is independent of the others, such that:

$$P(I^n | R_\alpha, \Gamma^{n-1} \dots \Gamma^0) = P_{R_\alpha}^\Gamma(I^n) = \prod_{\mathbf{x} \in R_\alpha} P_\alpha(I^n),$$

where $\alpha = \{\text{obj}, \text{bck}\}$.

For $\mathbf{x} \in \Gamma$, let $R_{\text{obj}}(\mathbf{x})$ and $R_{\text{bck}}(\mathbf{x})$ denote a neighborhood of \mathbf{x} in R_{obj}^Γ and R_{bck}^Γ respectively, such that $R_{\text{obj}}(\mathbf{x}) \in R_{\text{obj}}^\Gamma$ and $R_{\text{bck}}(\mathbf{x}) \in R_{\text{bck}}^\Gamma$. Based on these subregions, we can estimate P'_Γ using:

$$P'_\Gamma = \frac{\prod_{\mathbf{x}_1} \left[\overbrace{\prod_{\mathbf{x}_2} P_{R_{\text{obj}}^\Gamma}^\Gamma(I^n(\mathbf{x}_2))}^{\text{object likelihood}} \overbrace{\prod_{\mathbf{x}_3} P_{R_{\text{bck}}^\Gamma}^\Gamma(I^n(\mathbf{x}_3))}^{\text{background likelihood}} \overbrace{P_S^n(\mathbf{x}_1)}^{\text{shape likelihood}} \right]}{C}, \quad (5.6)$$

where $\mathbf{x}_1 \in \Gamma$, $\mathbf{x}_2 \in R_{\text{obj}}(\mathbf{x}_1)$, $\mathbf{x}_3 \in R_{\text{bck}}(\mathbf{x}_1)$ and P_S^n is a function of the contour variable \mathbf{x}_1 .

5.2 MAP Estimation

The maximum a posteriori estimate (MAP) of the object contour being tracked in the n^{th} frame, $\widehat{\Gamma}^n$, is found by maximizing the probability P'_Γ over the subsets $\Gamma \subset \Omega$, where Ω is the space of all possible object contours in the current frame, such that the object and the background regions are well separated.

Based on Equation 5.6, the MAP estimate of the object contour $\widehat{\Gamma}$ can be written in terms of the subregions defined by the contour. Thus, we have the following tracking scheme:

$$\widehat{\Gamma}^n = \arg \max_{\Gamma \subset \Omega} \prod_{\mathbf{x}_1} \left[\prod_{\mathbf{x}_2} P_{R_{\text{obj}}^\Gamma}(I^n(\mathbf{x}_2)) \prod_{\mathbf{x}_3} P_{R_{\text{bck}}^\Gamma}(I^n(\mathbf{x}_3)) P_S^n \right], \quad (5.7)$$

where \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 are as defined above. Note that the constant C is dropped due to the MAP estimation process.

In Equation 5.7, the band, $R_{\text{obj}}^\Gamma \cup R_{\text{bck}}^\Gamma$, around the hypothesized object contour, $\widehat{\Gamma}$, serves both as a *boundary constraint* (similar to boundary based active contour methods [CKS95, KWT88]) and as a *region constraint* (similar to [Man02, ZY96, PD02]) and generalizes the previously proposed contour based segmentation and tracking schemes into one framework.

The main advantages of the proposed tracking functional are:

- Noise and artifacts (holes inside the object) are not considered in the estimation, increasing robustness,
- The boundary-based and region-based methods are generalized into one framework,

- It allows object tracking using mobile cameras, and adapts to visual changes (due illumination, camera motion etc.) in the object and the background features.

5.2.1 Defining the Energy Functional

A convenient way of converting a MAP estimation to an energy minimization problem is by computing the *negative log-likelihood* of the probabilities. Thus, the tracking functional proposed in Equation 5.7 can be formulated as a minimization problem with the energy functional:

$$E(\Gamma) = \iint_{\mathbf{x}_1} \left[\underbrace{\iint_{\mathbf{x}_2} \Psi_{\text{obj}}(\mathbf{x}_2) d\mathbf{x}_2}_{E_A} + \underbrace{\iint_{\mathbf{x}_3} \Psi_{\text{bck}}(\mathbf{x}_3) d\mathbf{x}_3}_{E_B} - \log P_S^n \right] d\mathbf{x}_1, \quad (5.8)$$

where $\mathbf{x}_1 = (x_1, y_1)^T \in \Gamma$, $\mathbf{x}_2 = (x_2, y_2) \in R_{\text{obj}}(\mathbf{x}_1)^T$, $\mathbf{x}_3 = (x_3, y_3)^T \in R_{\text{bck}}(\mathbf{x}_1)$, and $\Psi_{\text{obj}}(\mathbf{x}) = -\log P_{R_{\text{obj}}}(I^n(\mathbf{x}))$, $\Psi_{\text{bck}}(\mathbf{x}) = -\log P_{R_{\text{bck}}}(I^n(\mathbf{x}))$.

The most important limitation of the energy functional given in Equation 5.8 is the lack of subregion definitions $R_{\text{obj}}(\mathbf{x})$ and $R_{\text{bck}}(\mathbf{x})$, and as a result the upper and lower bound of the surface integrals in Equation 5.8 are not defined. To overcome this problem, we introduce square subregions for $R_{\text{obj}}(\mathbf{x})$ and $R_{\text{bck}}(\mathbf{x})$:

$$R_\alpha(\mathbf{x}) = \{x, y \mid x \in [-m, m] \wedge y \in [-m, m]\},$$

where $\mathbf{x} \in \Gamma$ and $\alpha \in \{\text{obj}, \text{bck}\}$ and $R_\alpha(\Gamma) = R_{\text{obj}} \cup R_{\text{bck}}$. Construction of the sub-band around the contour using square subregions is presented in Figure 5.5b.

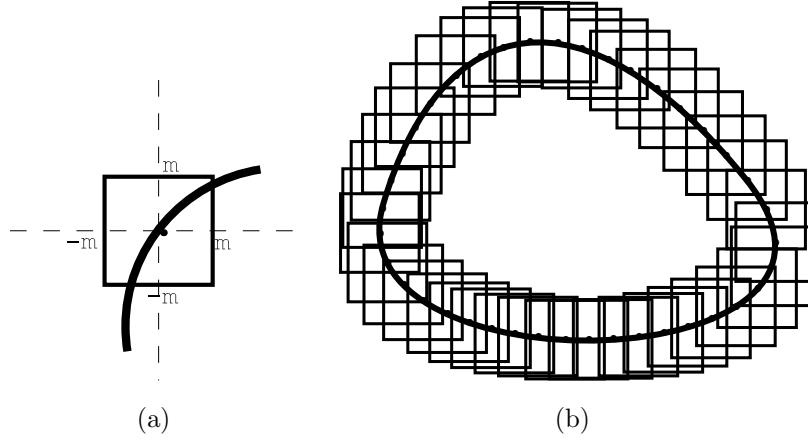


Figure 5.5: (a) Rectangular subregion, where bold line is the object contour and m is limit of the rectangle. (b) Band around the object contour defined by the rectangular subregions of (a).

A closer look at the square subregions centered around the contour, which is given presented in Figure 5.5a, shows that the membership definition is missing. Thus, if we replace the surface integrals $\iint_{\mathbf{x}_2}$ and $\iint_{\mathbf{x}_3}$ in Equation 5.8 with the integrals in the square region, i.e., $\int_{-m}^m \int_{-m}^m$, then the object and background membership is lost, due to the replacement of variables $\mathbf{x}_2 \in R_{\text{obj}}$ and $\mathbf{x}_3 \in R_{\text{bck}}$. To satisfy the requirement of the membership, we define an indicator function, $1_{\alpha}^{\Gamma}\{\mathbf{x} \in R_{\alpha}\}$, of the form:

$$1_{\alpha}^{\Gamma}\{\mathbf{x} \in R_{\alpha}\} = \begin{cases} 1 & \mathbf{x} \in R_{\alpha} \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha \in \{\text{obj}, \text{bck}\}$. The membership definition will become more clear in the following sections. The region defined by the square window centered around \mathbf{x}_1 can be represented

by:

$$\begin{aligned}x_i &= \tilde{x} + x_1, \\y_i &= \tilde{y} + y_1,\end{aligned}\tag{5.9}$$

for $i = 2, 3$ in Equation 5.8, $\tilde{x}, \tilde{y} \in [-m, m]$ and $x_i, y_i \in R_{\text{obj}}^\Gamma \cup R_{\text{bck}}^\Gamma$. Using this new definition of the contour band, E_A and E_B in Equation 5.8 becomes:

$$\begin{aligned}E_A(s) &= \iint_{-m}^m \Psi_{\text{obj}}(x_2, y_2) 1_{\text{obj}}^\Gamma\{x, y\} \frac{dx_2}{d\tilde{x}} d\tilde{x} \frac{dy_2}{d\tilde{y}} d\tilde{y}, \\E_B(s) &= \iint_{-m}^m \Psi_{\text{bck}}(x_3, y_3) 1_{\text{bck}}^\Gamma\{x, y\} \frac{dx_3}{d\tilde{x}} d\tilde{x} \frac{dy_3}{d\tilde{y}} d\tilde{y},\end{aligned}$$

where:

$$\frac{dx_2}{d\tilde{x}} = 1, \quad \frac{dy_2}{d\tilde{y}} = 1, \quad \frac{dx_3}{d\tilde{x}} = 1, \quad \frac{dy_3}{d\tilde{y}} = 1.$$

After doing necessary manipulations, the energy functional given in Equation 5.8 becomes:

$$\begin{aligned}E(\Gamma) &= - \iint_{\mathbf{x}_1} \overbrace{\iint_{-m}^m \log P_{R_{\text{obj}}}(I(\mathbf{x}_2)) 1_{\text{obj}}^\Gamma\{\mathbf{x}_2\} d\tilde{x} d\tilde{y}}^{\Phi_{\text{obj}}(\mathbf{x}_1) \Rightarrow \text{posteriori object log likelihood}} d\mathbf{x}_1 - \\&\quad \iint_{\mathbf{x}_1} \overbrace{\iint_{-m}^m \log P_{R_{\text{bck}}}(I(\mathbf{x}_3)) 1_{\text{bck}}^\Gamma\{\mathbf{x}_3\} d\tilde{x} d\tilde{y}}^{\Phi_{\text{bck}}(\mathbf{x}_1) \Rightarrow \text{posteriori background log likelihood}} d\mathbf{x}_1 - \\&\quad \iint_{\mathbf{x}_1} \overbrace{\log P_S^n}^{S(\mathbf{x}_1) \Rightarrow \text{shape}} d\mathbf{x}_1,\end{aligned}\tag{5.10}$$

where $\mathbf{x}_1 \in \Gamma$, $1_{\text{bck}}^\Gamma = 1 - 1_{\text{obj}}^\Gamma$ and $\mathbf{x}_2, \mathbf{x}_3$ are given in Equation 5.9.

5.2.1.1 Properties of the Energy Functional

The functional proposed in Equation 5.10 is general, and the functionals used by Mansouri [Man02], Caselles et al. [CKS95], Zhu and Yuille [ZY96], Paragios and Deriche, [PD02], etc. are special cases of this functional. If we drop the shape term from Equation 5.10, and perform the necessary modifications,

- Limiting the discriminant analysis to the pixels inside the region and setting the probabilities of Equation 5.10 to

$$P_\alpha(\mathbf{x}) = \max_{\mathbf{z}: \|\mathbf{z}\| \leq m} e^{-\frac{(I^{n-1}(\mathbf{x}) - I^n(\mathbf{x}+\mathbf{z}))^2}{2\sigma^2}},$$

where z defines the circular neighborhood of a pixel and dropping the plane integrals (due to max operation) results in the tracking scheme proposed in [Man02].

- Replacing the regional terms, Φ_{obj} and Φ_{bck} of Equation 5.10 with:

$$P_\alpha(\mathbf{x}) = e^{-\frac{|\Delta I|^2}{2\sigma^2}},$$

where $|\Delta I|$ is the image gradient. In addition, setting $m = 1$, which cancels the surface integral terms, the method reduces to the segmentation approach of [CKS95].

- The proposed objective function can be simplified to the two region case of [ZY96] by increasing m such that the square window covers the complete object and background regions. In this case, the front dependent subregions R_α^Γ of Equation 5.8 are changed to region terms R_α and the requirement of introducing extra derivations to deal with the involvement of the contour outlined in Section 5.2.1.1 can be discarded.

- The convex combination of boundary force (around a small neighborhood of the contour) and region force (similar to [ZY96]), which is proposed in [PD02], is implicitly obtained by the band size m and the membership function 1_α^Γ in Equation 5.8.

5.2.2 Minimizing the Tracking Functional

Object tracking using the Equation 5.10 requires that (starting from an initial estimate) the final contour, Γ , of the hypothesized object, R_{obj} has the minimum energy, $E(\Gamma)$. However, note that the energy functional given in Equation 5.8 includes fourth order integrals. In order to simplify the solution to our problem, we will reduce the order of integrals using *Green's Theorem*, which is a form of the *fundamental theorem of calculus* in the context of integrals over planar regions.

Theorem 1 *Green's Theorem:* *For a planar region R , $(P(x, y), Q(x, y))$ is any vector field with continuous first order derivatives, then:*

$$\iint_R \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy = \int_{\partial R} (P dx + Q dy),$$

where the boundary, ∂R , is traversed counterclockwise on its outside cycle, (and clockwise on any internal cycles).

Let us define a function F such that:

$$F(x, y) = \Phi_{\text{obj}}(x, y) + \Phi_{\text{bck}}(x, y) + S(x, y) = \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}, \quad (5.11)$$

where Φ_{obj} , Φ_{bck} and S are defined in Equation 5.10. Intuitive Q and P functions that satisfies the requirements in Equation 5.11 are [ZY96]:

$$Q(x, y) = \frac{1}{2} \int_0^x (\Phi_{\text{obj}}(t, y) + \Phi_{\text{bck}}(t, y) + S(t, y)) dt,$$

$$P(x, y) = - \frac{1}{2} \int_0^y (\Phi_{\text{obj}}(x, t) + \Phi_{\text{bck}}(x, t) + S(x, t)) dt.$$

At this point, in order to use conventional minimization techniques for parametric functions, let's represent the contour by its parametric form:

$$\Gamma : \begin{cases} x = f(s) \\ y = g(s) \end{cases},$$

where $0 \leq s \leq l$, l is the arc-length of Γ , f and g are the parametric curve functions, and x and y are the variables of the region boundary, ∂R . Changing the variables of the integral around the line given in Theorem 1, we have:

$$\int_{\partial R} (P dx + Q dy) = \int_0^l (P \dot{x} + Q \dot{y}) ds,$$

where $\dot{x} = \frac{\partial x}{\partial s}$ and $\dot{y} = \frac{\partial y}{\partial s}$. Let

$$L(x, \dot{x}, y, \dot{y}) = P(x, y) \dot{x} + Q(x, y) \dot{y}.$$

Based on these derivations the contour energy functional becomes:

$$\begin{aligned} E(\Gamma) &= \int_0^l L(x, \dot{x}, y, \dot{y}) ds \\ &= -\frac{1}{2} \int_0^l \int_0^y (\Phi_{\text{obj}}(x, t) + \Phi_{\text{bck}}(x, t) + S(x, t)) dt \dot{x} ds + \\ &\quad \frac{1}{2} \int_0^l \int_0^x (\Phi_{\text{obj}}(t, y) + \Phi_{\text{bck}}(t, y) + S(t, y)) dt \dot{y} ds. \end{aligned} \tag{5.12}$$

Note that, the order of integrals reduced from four to three. The first order necessary condition in regard to minimizing the energy functional given in Equation 5.12 is to compute the gradient of the functional by the Euler-Lagrange equation:

$$\frac{\delta E}{\delta x} = \frac{\partial L}{\partial x} - \frac{d}{ds} \frac{\partial L}{\partial \dot{x}}, \quad (5.13)$$

$$\frac{\delta E}{\delta y} = \frac{\partial L}{\partial y} - \frac{d}{ds} \frac{\partial L}{\partial \dot{y}}. \quad (5.14)$$

Substituting the derivatives:

$$\begin{aligned} \frac{\partial L}{\partial x} &= \frac{\partial P}{\partial x} \dot{x} + \frac{\partial Q}{\partial x} \dot{y}, & \frac{\partial L}{\partial \dot{x}} &= P(x, y), \\ \frac{\partial L}{\partial y} &= \frac{\partial P}{\partial y} \dot{x} + \frac{\partial Q}{\partial y} \dot{y}, & \frac{\partial L}{\partial \dot{y}} &= Q(x, y), \end{aligned}$$

and

$$\begin{aligned} \frac{d}{ds} \frac{\partial L}{\partial \dot{x}} &= \frac{d}{ds} P(x(s), y(s)) = \frac{\partial P}{\partial x} \dot{x} + \frac{\partial P}{\partial y} \dot{y}, \\ \frac{d}{ds} \frac{\partial L}{\partial \dot{y}} &= \frac{d}{ds} Q(x(s), y(s)) = \frac{\partial Q}{\partial x} \dot{x} + \frac{\partial Q}{\partial y} \dot{y}, \end{aligned}$$

into the Euler-Lagrange Equations 5.13 and 5.14, we get:

$$\begin{aligned} \frac{\delta E}{\delta x} &= \frac{\partial Q}{\partial x} \dot{y} - \frac{\partial P}{\partial y} \dot{y}, \\ \frac{\delta E}{\delta y} &= -\frac{\partial Q}{\partial x} \dot{x} + \frac{\partial P}{\partial y} \dot{x}. \end{aligned}$$

The Euler-Lagrange equations given above are related to the functional F given in Equation 5.11, such that:

$$\frac{\delta E}{\delta x} = (\Phi_{\text{obj}} + \Phi_{\text{bck}} + S) \dot{y}, \quad (5.15)$$

$$\frac{\delta E}{\delta y} = -(\Phi_{\text{obj}} + \Phi_{\text{bck}} + S) \dot{x}, \quad (5.16)$$

The Equations given in 5.15 and 5.16 are a differential system of equations that include integral terms. Formally, they are system of partial integro-differential equations of order one [AKZ00]:

$$\frac{\delta E}{\delta x} = \left(\iint_{-m}^m \Psi_{\text{obj}}(\mathbf{x}) 1_{\text{obj}}^{\Gamma}\{\mathbf{x}\} + \Psi_{\text{bck}}(\mathbf{x}) 1_{\text{bck}}^{\Gamma}\{\mathbf{x}\} d\mathbf{x} + S \right) \dot{y}, \quad (5.17)$$

$$\frac{\delta E}{\delta y} = - \left(\iint_{-m}^m \Psi_{\text{obj}}(\mathbf{x}) 1_{\text{obj}}^{\Gamma}\{\mathbf{x}\} + \Psi_{\text{bck}}(\mathbf{x}) 1_{\text{bck}}^{\Gamma}\{\mathbf{x}\} d\mathbf{x} + S \right) \dot{x}. \quad (5.18)$$

Let $\vec{v} = (x, y)^T$, the normal at \vec{v} is $\vec{n} = (\dot{y}, -\dot{x})^T$. In Section 5.1, we defined the contour to be a counter-clockwise curve for the object region and a clockwise curve for the background region (see Figure 5.2). Thus, the normal of the object region \vec{n}_{obj} is in the opposite direction of \vec{n}_{bck} :

$$\vec{n}_{\text{obj}} = -\vec{n}_{\text{bck}}.$$

Using these, the system of integro differential equations given in 5.17 and 5.18 becomes:

$$\frac{\delta E}{\delta \vec{v}} = \left(\iint_{-m}^m \Psi_{\text{obj}}(\mathbf{x}) 1_{\text{obj}}^{\Gamma}\{\mathbf{x}\} - \Psi_{\text{bck}}(\mathbf{x}) 1_{\text{bck}}^{\Gamma}\{\mathbf{x}\} d\mathbf{x} + S \right) \vec{n}_{\text{obj}}.$$

In the following discussion, without loss of generality, we will refer to the object normal \vec{n}_{obj} as \vec{n} . From the above, the optimum direction for minimizing the energy functional of a spatial contour is obviously the normal direction. However, it should be noted that partial integro-differential equations are not very common and hard to solve in general, especially the way derived above by the introduction of a square neighborhood definition. Although the convergence issue needs rigorous analysis, due to the changing membership regions, a practical approach to show its convergence is to assume that the rectangle centered on the contour has equal object and background regions on all the contour positions. Thus, the

planar integral can be dropped and the system of equations can be transformed into a second order differential equation, whose convergence has been well studied.

5.3 Contour Representation and Evolution

The contour of an object, which undergoes non-rigid deformations, can be represented in various ways including [Chu02]:

- *Parametric form:* is the most common and the most limited approach to defining various shapes. It cannot change topology and may result in problems during evolution. An intuitive example which allows scale changes (one degree of freedom) is the parametric form of the contours

$$x = r \cos s, \qquad y = r \sin s.$$

- *Marker-string method:* is usually used for active contour framework, it was first introduced in the fluid dynamics discipline. The idea is to represent the contour by a fixed number of control points. The relations between the points are usually defined through splines. This representation does not allow topology changes and causes problems if two control points evolve to the same position.
- *Volume fluid method:* are the most common contour representation in the fluid dynamics field. For the two dimensional case, the space is divided into subspaces of equal area

and the closed contour is represented such that, inside the contour, the subspaces are filled with fluid, whereas outside the contour, the subspaces are empty. For subspaces, which the contour partially intersects, the portion of the subspace is filled to reflect the amount of subregion belonging to the inside of the contour.

- *Level set:* is a new tool to represent the contours implicitly (similar to volume fluid method). Level set representation is easier to implement, and in contrast to volume fluid methods, contour properties such as curvature and normal can be computed from the representation. We will discuss the level sets in more detail in the next section.

Here, we chose the level set level set representation for its simplicity and stability for representing the contour of the object.

5.3.1 Level Set Representation

Level sets implicitly represent a contour on a grid, $\phi : R^2 \times R \rightarrow R^1$, which is usually called the *front*. The position of the contour is defined by the values on the grid. Formally the contour is referred as the *0-level*, such that $\phi(\Gamma) = 0$. The inside and outside of the contour take opposite signs:

$$\text{sign}\{\phi(R_{\text{in}}^{\Gamma})\} \times \text{sign}\{\phi(R_{\text{out}}^{\Gamma})\} = -1$$

Since grid values are not always zero, practically, the zero crossings on the grid is taken to find the position of the contour. A sample level set representation of a contour is shown in

Figure 5.6a. Note that the contour is not necessarily on the grid (if we assume that the grid is the image pixels then we obtain sub-pixel accuracy).

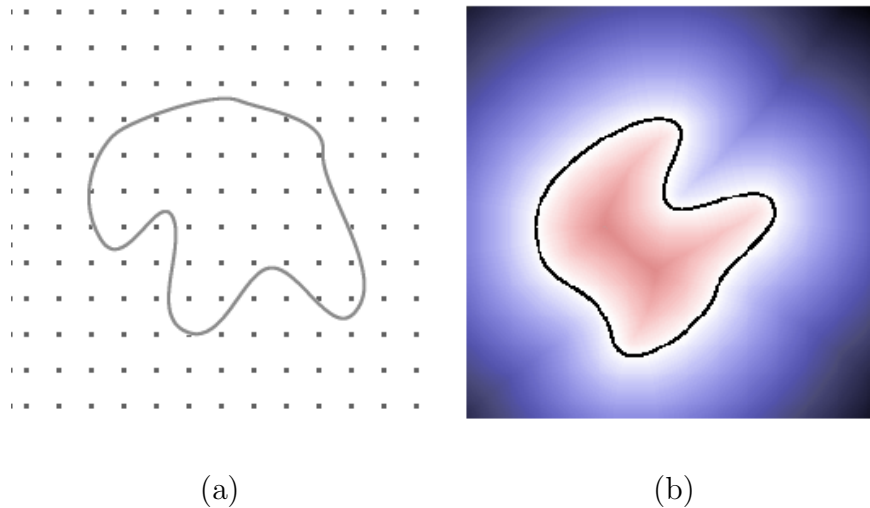


Figure 5.6: (a) Contour on a level set grid. (b) Distance transformation with contour superimposed.

A common measure used as grid values in a level set representation are distance from the contour, which can be obtained by a distance transformation [Cui99]. In Figure 5.6b, we show the distance transformation of a given contour generated using the fast algorithm proposed in [Cui99]. The contour in this image is the 0-level and the rest of the pixels represent the other levels, i.e., first, second, third, etc.

An evolving contour in level set representation is defined by introducing a time parameter, t , in the formulation, such that $\phi(\Gamma(s, t), t) = 0$, where s is the parameter of the contour and $\Gamma(s, t) = \{x(s, t), y(s, t)\}$. Evolving the contour corresponds to updating the level set

function, ϕ , which defines new zero crossings. The level set evolution is defined by

$$\frac{d\phi}{dt} = \phi_t + \nabla\phi(\Gamma(s, t), t) \frac{\partial\Gamma}{\partial t}.$$

Since we are working on the 0-level set, $\phi(\Gamma) = 0$, we have:

$$\begin{aligned} 0 &= \frac{\phi_t}{|\nabla\phi|} + \frac{\overbrace{\nabla\phi(\Gamma(s, t), t)}^{\text{normal vector} \Rightarrow \vec{n}}}{|\nabla\phi|} \Gamma', \\ \phi_t &= |\nabla\phi| \underbrace{\vec{n} \Gamma'}_{\text{velocity} \Rightarrow \vec{v}}. \end{aligned} \tag{5.19}$$

We can interpret Equation 5.19 as the evolution of the contour in its normal direction with speed F . In Figure 5.7, we show the evolution of a curve when F is set to the curvature of the curve. Note that the evolution of the contour tends to reduce the curvature, such that after some iterations the shape becomes a circle, and eventually collapses to a point.

5.3.2 Relating Level Sets with Energy Minimization

The energy functional defined in Section 5.2.1.1, can be related to evolving the contour by introducing one more dimension, t , in the process. Thus the contour becomes a function that has two parameters, $\Gamma(s, t)$. Minimizing the related energy functional given in Equation 5.10 can be converted to minimizing the energy temporally by $\frac{\partial E}{\partial t}$:

$$\frac{\partial E(\Gamma(s, t))}{\partial t} = \frac{\partial E}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial E}{\partial y} \frac{\partial y}{\partial t}. \tag{5.20}$$

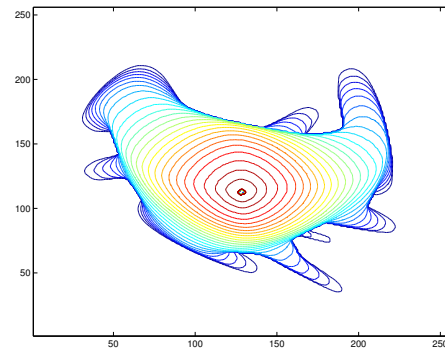
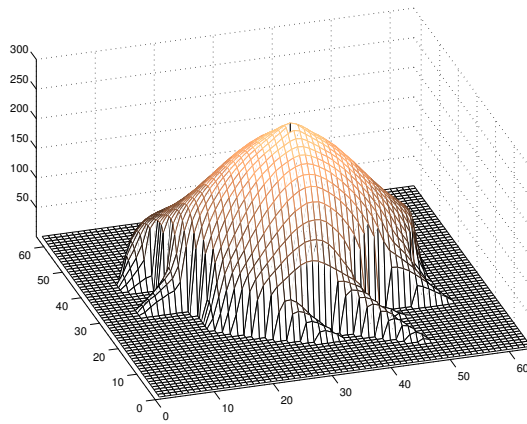
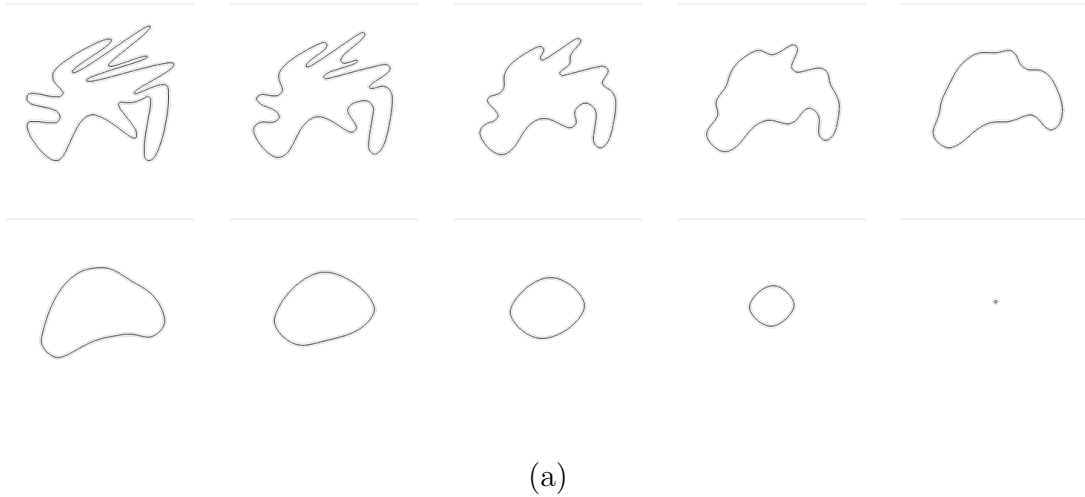


Figure 5.7: Evolving contour under curvature flow. (a) 0-levels at various time slices. (b) The surface constructed by curvature based contour evolution. Note that cutting the surface at various levels will give the evolving contours given in (a). (c) Contour plot of the surface given in (b).

In Equation 5.20, partial derivatives, $\frac{\partial E}{\partial x}$ and $\frac{\partial E}{\partial y}$ are derived in Equations 5.17 and 5.18.

Rearranging terms, we have

$$\frac{\partial E(\Gamma(s, t))}{\partial t} = \left(\frac{\partial E}{\partial x} \frac{\partial E}{\partial y} \right) \begin{pmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{pmatrix}, \quad (5.21)$$

where the first part of the equation corresponds to the combined Euler-Lagrange equations:

$$\begin{pmatrix} \frac{\partial E}{\partial x} \\ \frac{\partial E}{\partial y} \end{pmatrix} = \frac{\partial E}{\partial \vec{v}} \vec{n}.$$

Plugging this into Equation 5.21, we have

$$\frac{\partial E(\Gamma(s, t))}{\partial t} = (\Phi_{\text{obj}} - \Phi_{\text{bck}} + S) \left(\frac{\partial \vec{v}}{\partial t} \right)^T \vec{n}, \quad (5.22)$$

where $\left(\frac{\partial \vec{v}}{\partial t} \right) = \left[\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t} \right]^T$ and $\Phi_\alpha : \alpha \in \{\text{obj}, \text{bck}\}$ is defined in Equation 5.10. Minimizing the energy according to the evolving contour parameter t is then related to level set evolution function given in Equation 5.19 by the following speed function:

$$F = \iint_{-m}^m (\Psi_{\text{obj}}(\mathbf{x}) 1_{\text{obj}}^\Gamma\{\mathbf{x}\} - \Psi_{\text{bck}}(\mathbf{x}) 1_{\text{bck}}^\Gamma\{\mathbf{x}\}) d\mathbf{x} + S.$$

The level set representation outlined in Section 5.3.1 is a grid-based representation and is discrete in nature. Thus the speed function given above can simply be converted to a discrete function by:

$$\begin{aligned} F_{x,y} = & - \sum_{i=-m}^m \sum_{j=-m}^m \log P_{R_{\text{obj}}}(I_{\mathbf{x}'}) 1_{\text{obj}}^\Gamma\{(\mathbf{x}')\} + \\ & \sum_{i=-m}^m \sum_{j=-m}^m \log P_{R_{\text{bck}}}(I_{\mathbf{x}'}) 1_{\text{bck}}^\Gamma\{(\mathbf{x}')\} - \\ & S(x, y), \end{aligned} \quad (5.23)$$

where $\mathbf{x}' = (x + i, y + j)$. The negative and the positive terms that have double summations in Equation 5.23 increase or decrease the value of the level set grids, $\phi(x, y)$, which in turn changes the position of the zero crossings. We can interpret these shrinking and expanding forces as:

- When the boundary hypothesis for the pixel \mathbf{x}_r is correct, the speed, and thus the motion, of the contour pixels will be ≈ 0 .
- If the object contour is not correct, the background (object) likelihood of the front pixel will be higher and the speed function becomes negative (positive).

In our implementation, an object is considered to be either occluded or unoccluded during tracking. Since there is no prior object information, and we build our appearance models on-line, we are not considering the shape component in Equation 5.23, i.e., $S(x, y) = 0$, during the contour evolution unless occlusion occurs, where the object shape is dramatically distorted. Besides, using the shape constraint during tracking without occlusion is computationally expensive.

5.4 Tracking During Occlusion

So far, we have not dealt with occlusion, and only considered the appearance models using visual features such as color and texture. However, visual features are not observed during occlusion. Missing observations in such cases degrade the performance of the object tracking. In this section, we discuss our approach to detecting the occlusions, label the occluder and the occludee and define an on-line shape prior based on the level set representation, which can account for the missing observations and track the contours of the objects.

5.4.1 Occlusion Detection

The implicit definition of an object contour in a level set representation is given by the zero crossings on the grid of the level set function $\phi(\Gamma, t)$, where the grid values can be defined by the Euclidean distance from the contour. Let there be N objects with N level set functions $\phi_i : i \leq N$, having the object area A_i^t at frame t . The distance $D_{i,j}$ from the object O_i to the object O_j can be computed by:

$$D_{i,j} = \arg \min \phi_i(\Gamma_j(\mathbf{x})),$$

where $i \neq j$. Given the average area:

$$A_i^{avg} = \tau A_{i-1}^{avg} + (1 - \tau) A_i^t.$$

which is computed over time, the occlusion decision is made based on both $D_{i,j}$ and the ratio of A_i^t to A_i^{avg} using:

$$Occ_{i,j} = \underbrace{\frac{1}{e^{-|D_{i,j}|} + 1}}_{\text{measure of closeness to } O_j} \times \underbrace{\frac{A_i^t}{A_i^{avg}}}_{\text{ratio of the area}}. \quad (5.24)$$

To give more insight to Equation 5.24, let's consider the following scenarios:

- When objects O_i and O_j are close to each other such that $D_{i,j} \approx 0$, the first term becomes 0.5 (lower bound). Based on the value of the second term, object O_i is labeled as the *occludee* and object O_j is labeled as the *occluder*. For the occludee (O_i), shape reconstruction is started by evolving using the scheme described in the next section.

- If the objects O_i and O_j are far apart, such that $D_{i,j} \gg 0$, the first term becomes 1 (upper bound). At this point the algorithm is aware that there is no occlusion. However, based on the second term (ratio of the areas) a significant decrease in the object area can be detected. Decrease of the object area occurs for objects that have visual features very similar to their backgrounds, where the contour evolution becomes unpredictable. In this case, although there is neither an occluder nor an occludee, we reconstruct the shape of O_i using the scheme given in the next section.

The decision process described above can be understood by looking at Figure 5.8, in which two synthetic objects (occluders) are occluding the tennis player (occludee). In Figure 5.8a, we show the first frame of the sequence, (b) shows the contours of three objects, which are superimposed on the objects, labeled with different contour colors. In Figure 5.8c, d, e, extracted objects for the first frame are shown. Occlusion occurs in the second frame of the sequence (Figure 5.8f). Using the visual features (color and texture), the tracking method described in Section 5.3.2 finds the object contours which is shown in Figure 5.8g. Figure 5.8h, i, j shows the extracted objects. At this point, the distances between the player and the other two objects are $D_{\text{player, ellipse}} \approx 0$ and $D_{\text{player, rectangle}} \approx 0$. The ratio of the player area and its average area computed over time is $\frac{A_{\text{player}}}{A_{\text{player}}^{\text{avg}}} < 0.5$. Thus, Equation. 5.24 is $Occ_{\text{player,rectangle}} < .25$ and $Occ_{\text{player,ellipse}} < .25$, which conclude that the player is occluded by both of the synthetic objects.

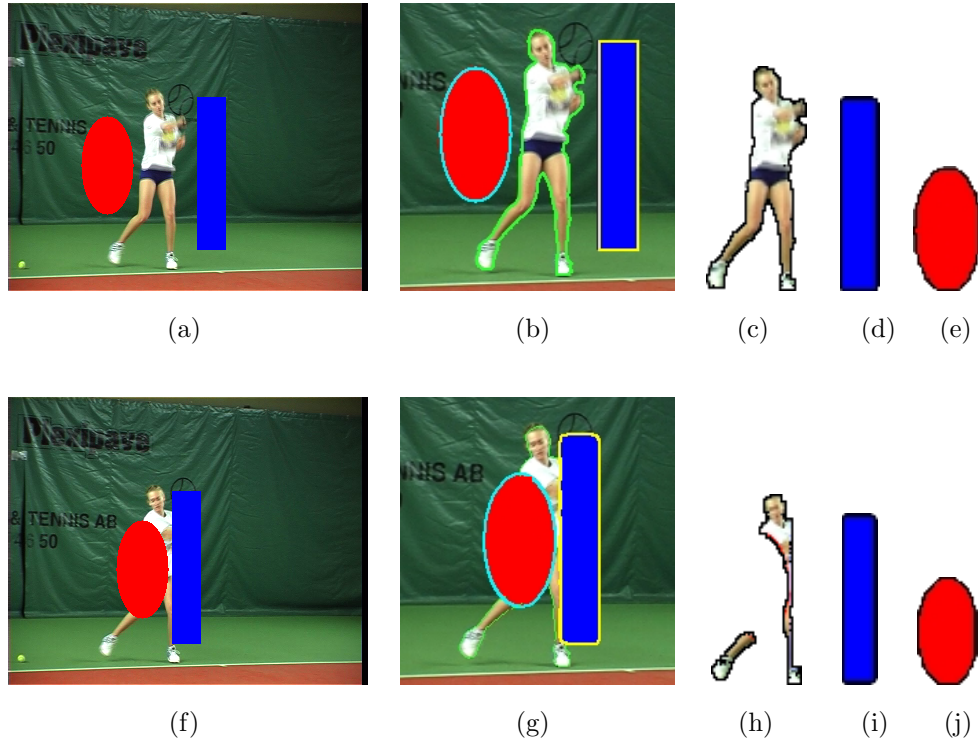


Figure 5.8: Sequence of frames with two synthetic objects (red and blue) occluding the tennis player. (a) First frame of the sequence. (b) Initial object contours super imposed on the objects. Extracted (c) player, (d) rectangular, and (e) circular objects. (f) Second frame of the sequence. (g) Tracking results using the visible features (contours are superimposed on the objects). Extracted (h) player, (i) rectangular, and (j) circular objects. Note the missing regions in (h) due to occlusion.

Next, we will discuss how the algorithm estimates the contour of the missing regions, which is shown in Figure 5.8h.

5.4.2 Estimating the Occluded Object Contour

Objects usually undergo non-rigid deformations resulting in changes of shape, S , and area, A . To keep track of this deformation, we represent each object, O_i , with a dense level set, ϕ'_i , which is obtained by increasing the number of grid points as shown in Figure 5.9a, b. However, in contrast to ϕ_i , the outside region of the contour in ϕ'_i is set to zero (Figure 5.9c).

Let the deformations be represented by dense level sets, $\phi'_i(\Gamma^1, 1)$, $\phi'_i(\Gamma^2, 2), \dots, \phi'_i(\Gamma^t, t)$, in the spatio-temporal domain (Figure 5.9c). Ideally, the shape, S_i , for the object, O_i , changes gradually, resulting in a small variation in ϕ'_i over time. For each grid point (k, l) , we model the change of values by a single Gaussian, $G_{\phi'}(k, l)$. The models are updated until an occlusion is detected based on the criteria given in Equation 5.24. In presence of an occlusion at time j for O_i , the level set function $\phi_i(\Gamma^j, j)$ (with fewer grid points) is obtained by evolving the contour using the visible features, which leads to missing observations (object regions that are missing as shown in Figure 5.8h). In this case, for the missing regions, the contour will be evolved to maximize the likelihood of observing the modeled shape, $G_{\phi'}$, by setting Φ_{obj} and Φ_{bck} to 0 in Equation 5.23 and computing:

$$F(k, l) = S(k', l') = G_{\phi'}(k', l') = \frac{1}{\sqrt{2\pi}\sigma_{k,l}} e^{-\frac{(\phi'_{k',l'} - \mu_{k',l'})^2}{\sigma_{k,l}^2}}, \quad (5.25)$$

where (k, l) is the grid position in the spatial domain, and (k', l') is the corresponding grid position in the dense level set ϕ' , and $\mu_{k,l}$, $\sigma_{k,l}$ are the Gaussian model parameters. Due to the initialization of the outside region as 0, the speed computed using Equation 5.25 will be

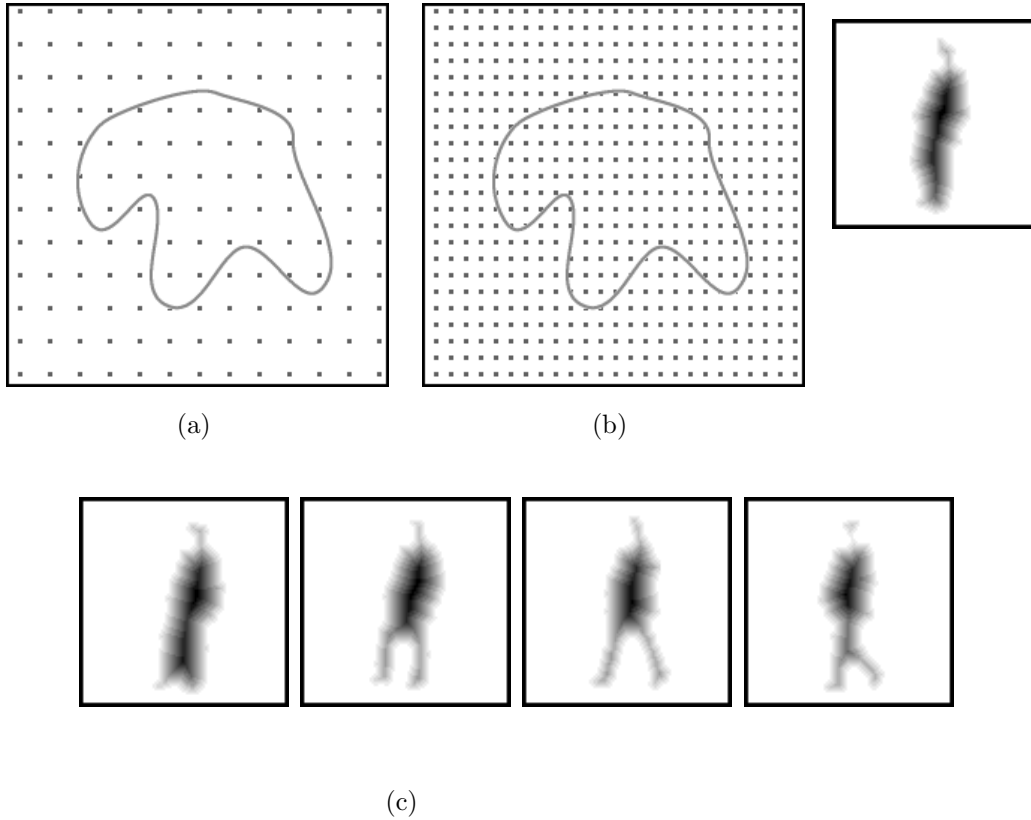


Figure 5.9: a) Contour represented on the spatial grid points, b) increasing the number of grids for the contour shown in (a), (c) consecutive level sets, ϕ'_i , from a walking person sequence with a coarser grid.

just an expansion force to fill in the missing observations during tracking with or without occlusion.

In Figure 5.10, we present the steps of the contour estimation algorithm for the occluded object regions. Figure 5.10a shows the object contours that are estimated using the evolution function given in Equation 5.23, in part (b), we show the estimated object contours of the

objects after the estimation process using Equation 5.25. Note that the missing regions are in Figure 5.10a are correctly recovered and the extracted objects are shown in Figure 5.10c, d and e. In Figure 5.10f, we display selected frames from the sequence, which demonstrates the robustness of the method. The second frame in Figure 5.10f has multiple occlusions: the ellipsoidal object is occluding both the tennis player and the rectangular object, while the rectangular object is occluding the tennis player. The proposed method successfully handled the multiple object occlusion. In Section 5.5, more results including full object occlusion are presented.

5.5 Experimental Results

To demonstrate the performance of the proposed contour tracking approach, we have tested our algorithm on various real sequences captured using stationary and mobile infrared (IR) and electro-optical (EO) cameras. Some of the test sequences are standard sequences used by various researchers in the field. The results we obtained are in general excellent and the contours of the objects are very tight. During the tracking, model priors for the objects are computed on-line by reevaluating the change in the object and background features. The contour evolution is implemented using the narrow band level set method, where Equation 5.23 is used as the speed function [Set99]. The algorithm is initialized with boundaries of objects in the first frame. The selection of m (the limits of the square region in

Equation 5.23) is not sequence dependent and is fixed to 6 for all sequences. In contrast to [Man02], magnitude of motion is not limited. For the video sequences, please visit: http://www.cs.ucf.edu/~vision/projects/contour_tracking.

5.5.1 Single Object Sequences

Tennis Player Sequence Figure 5.11 shows the tracking results on the standard tennis player sequence. In the sequence, the camera pans and tilts slowly as the player performs various strokes. Usually the visual features of the background and the player remains the same (especially the color, which is distinctive) and the contour of the player is perfectly tracked throughout the sequence (even the pony tail of the player is tracked!). Note that the racket is not tracked due to the motion blur resulting from fast racket motion. These results are probably the best tracking results compared to other recent work [HE02] [KS01].

Walking Person Sequence To demonstrate the robustness of the method for a mobile camera, we tested the proposed method on a challenging video sequence where the camera, zooms, pans, and tilts as shown in Figure 5.12. The visual features both for the person and the background vary as the person walks and sometimes the background model and the object model become very close in both color and texture. As can be observed from the sequence, sometimes the color or texture alone were adequate, however at other times

(toward the end of the sequence) both features contributed to the tracking. Throughout the sequence, the contour of the person is perfectly tracked and there were no miss-positioned contour points.

Surveillance Camera Person Walking Sequence In Figure 5.13, we present contour tracking results of a walking person sequence acquired from a low quality surveillance camera located in downtown Orlando. The color is distinctive enough to track the person. In particular, the background model is generated and updated such that the shadow of the person is a part of the model. Thus the shadow of the person is naturally left as a part of the background (such that it is outside the contour). The proposed method produced very tight object contour throughout the sequence.

Walking Person Sequence In Figure 5.14, contour tracking results for another real sequence is shown. The contour of the person is perfectly tracked, while the method adapts to the variation of the color and texture of the background. For instance at the beginning (first frame), the model generated for the background is different from the model generated toward the middle of the sequence (third frame).

5.5.2 VIVID Dataset

The VIVID dataset is composed of sequences acquired from aerial vehicles. The sequences are usually of low quality and the objects do not have very unique features to identify the correct object contour. Both the color and texture contribute to the tracking in all these sequences.

Passing EO: In Figure 5.15, we present contour tracking results for the two object case. The objects in the sequence are very small and hard to track. In the first frame we manually initialized and labeled the objects and for the rest of the sequence the labels and object contours are preserved. At some points of the tracking due to very low object resolution, some parts of the contour are misplaced but overall the contour is always on the objects.

Tank Sequence: The sequence given in Figure 5.16 is challenging for contour tracking since the object and background color and texture models become very similar during tracking also the models need to be correctly updated to adapt the changing features. Throughout the sequence, our method successfully tracked the object contour. As seen from the figure, at some points during the tracking the object contour is expanded incorrectly which does not reflect the correct object contour, This is due to the limited set of features used in tracking (color and texture).

Ural Sequence: In Figure 5.17, tracking results for a very long VIVID sequence is presented. As seen from the figure, two different poses of the vehicle are observed due to the motions of the aerial and ground vehicles. As observed from the 3rd image, a tail like contour, which does not belong to the object, is expanded. Although very easy to erase by a simple post processing (such as keeping only the biggest connected component) we left it to display the limitations of the features used for tracking. Besides this problem, the contour of the object is tracked reasonably well.

5.5.3 Infrared Sequences

In general, IR sensors produce very low quality images. Closing IR video (where the sensor is mounted on an aerial vehicle), in particular are highly effected by atmospheric conditions. We have tested our algorithm on various sequences from the AMCOM dataset, in which the intensity values of the objects and the background change rapidly even in consecutive frames, which requires a highly adaptive feature modeling mechanism. Figure 5.18 presents three different sequences. In all three sequences it should be noted that the brightness of the objects and backgrounds changes dramatically from beginning to end, and our method successfully generates statistical models for very small targets, (10 to 15 pixels in area). Notice the blurred boundary between the objects and the backgrounds in the first couple of frames shown in Figures 5.18a, b, c. Especially in Figure 5.18a, the window of the vehicle

(dark region) toward the end of the sequence is tracked as a part of the vehicle (besides its similarity to the background). These particular sequences will result in over segmented object regions if tried with the conventional active contour methods [CKS95, CS97, KWT88, PD02]. Methods given in [BBA00], [BSR00] and [PD00] will not work at all because they rely on static background and/or small object motion from frame to frame. In addition, due to the lack of repetitive texture in most part of the sequences both [JFE01] and [PD02] will fail to track the objects.

5.5.4 Occluding Object Sequences

In Figures 5.19 and 5.20, we demonstrate the ability of our approach to track objects under occlusion as described in Section 5.4. Both of the sequences demonstrate the performance under complete occlusion, where visual features do not provide any information. Most of the transformation-based object tracking approaches including [PD00, CRM00, JFE01, YSS03, Man02] which do not have a way to track the objects under occlusion will fail for both of these sequences.

Dancer Sequence: The dancer sequence (Figure 5.19) is a very tough sequence for generating a consistent shape model due to the rapid deformation around the skirts of both dancers. The first row in the figure displays the frames with contours superimposed, while

the second row shows the extracted dancers before, during and after occlusion. Our method correctly labelled the occluder (red dressed dancer) and the occludee (white dressed dancer) and correctly estimated the occluded (white dressed) dancer's position (depicted as *Dancer B* in the second row of the figure).

Occluding Persons Sequence: The occluding persons sequence (Figure 5.20) also demonstrates the robustness of both the tracking and the estimation of missing object regions. Notice the white regions inside *Person A* in the second row of Figure 5.20, especially during complete occlusion. Person A's contour is correctly estimated and both persons are correctly tracked before, during and after the occlusion. Another important point in this sequence is that the ambiguous regions (such as the heads of both persons) are correctly located and tracked during and after the occlusion. To the best of our knowledge, almost no tracking method in current literature can perform tracking with this much accuracy and estimate the correct contour of the objects during occlusion.

5.6 Conclusions

We proposed a contour based nonrigid object tracking method. Along with color and texture models generated for the object and the background regions, our method maintains a shape

prior for recovering occluded object parts during the occlusion. The shape priors encode the motion of the object and are built online. The energy functional is derived using a Bayesian framework and is evaluated around the contour to suppress visual artifacts and to increase numerical stability. We minimized the energy in the gradient descent direction, which in turn maximizes the posteriori contour probability. The results presented show the robust tracking performance with occlusion in video acquired from moving cameras.

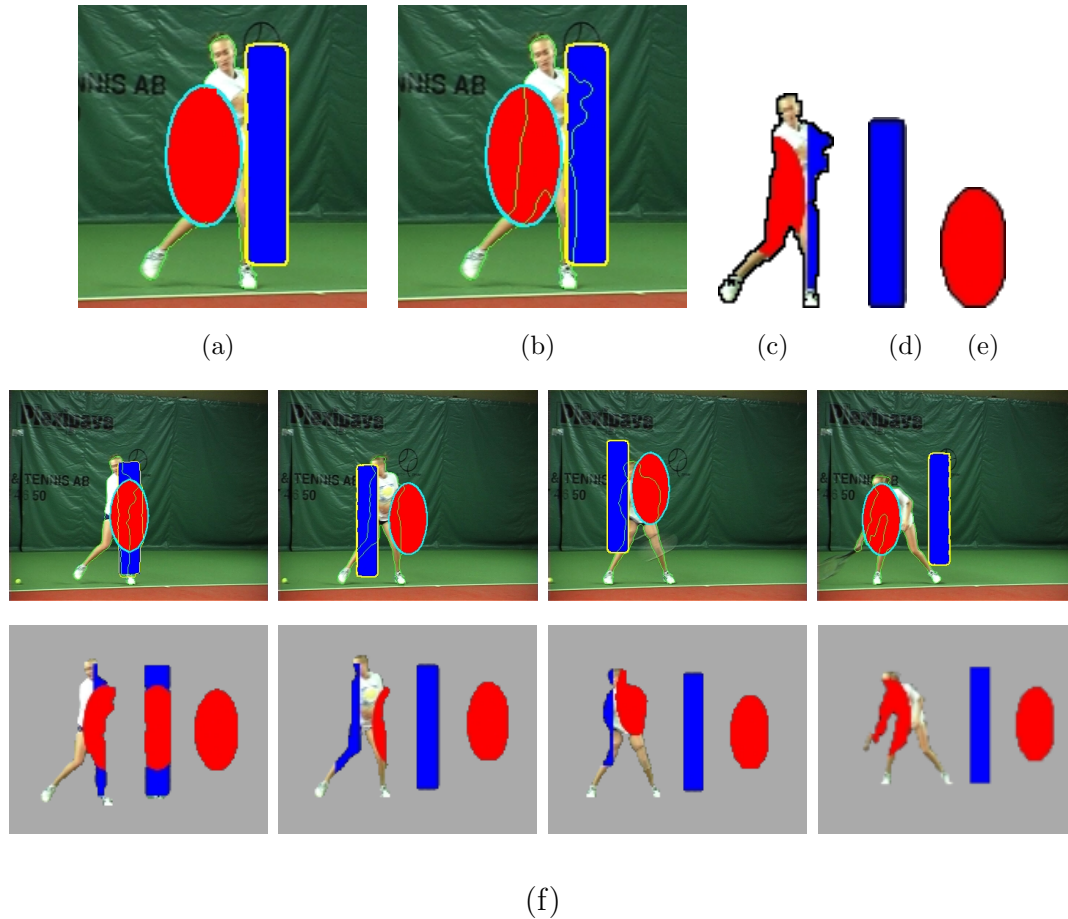


Figure 5.10: Synthetic occlusion sequence. (a) Tracking result using visible features (given in Figure 5.8g). (b) Tracking result using the occlusion handling method given in Equation 5.25 (contours superimposed). Note that the occludee is marked with a thin contour, whereas the occluders are marked with thick contours. Extracted (c) player, (d) rectangle and (e) ellipse. Note that the missing contour regions in Figure 5.8h are correctly estimated. (f) Selected frames from the complete sequence. The first row shows the contours superimposed and the second row shows extracted objects. Note that multiple object occlusion in the second frame of (f) is correctly handled.



Figure 5.11: Tracking results for the tennis player sequence acquired using a mobile camera.

We recommend viewing the results in color.

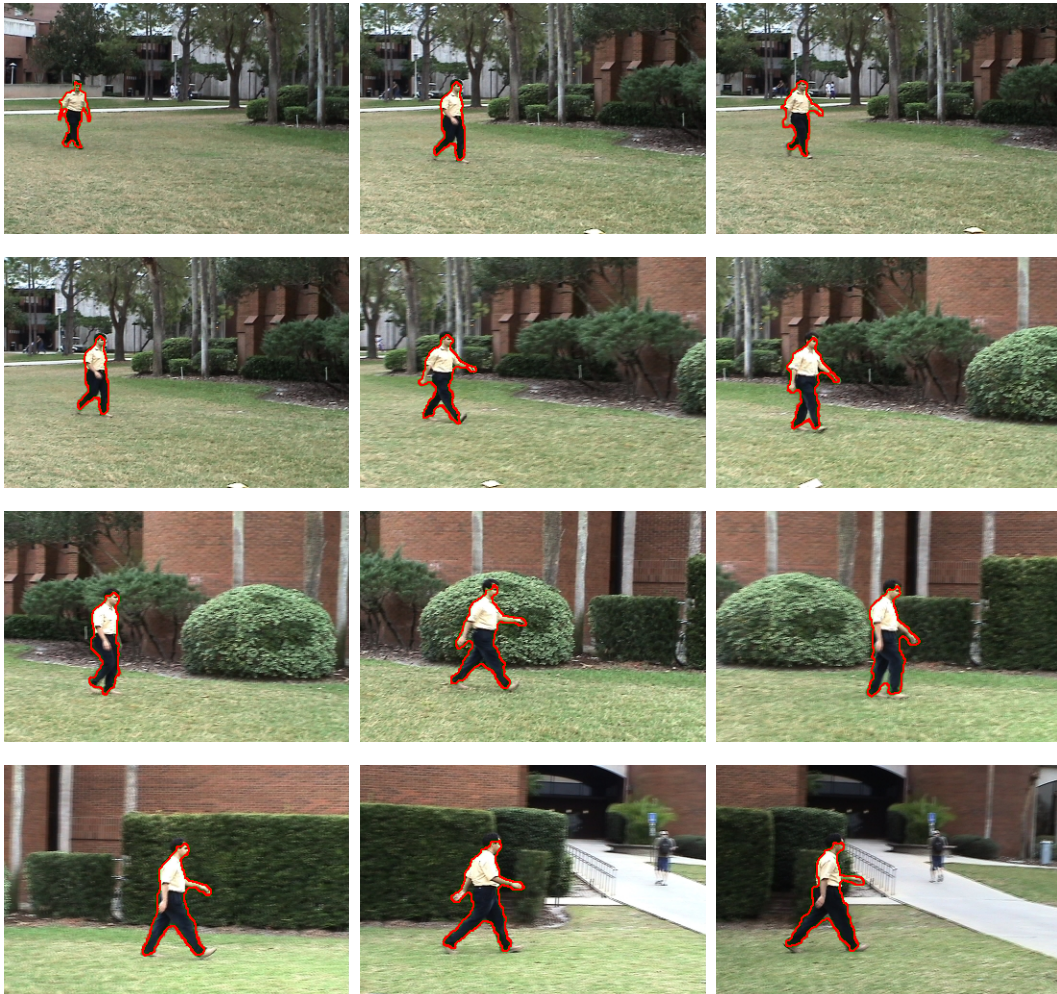


Figure 5.12: Contour tracking results for a walking person sequence, in which the visual features both for the foreground and the background change. We recommend viewing the results in color.



Figure 5.13: Contour tracking results for a sequence acquired from a stationary surveillance camera.



Figure 5.14: Contour tracking results for walking person sequence acquired using mobile camera.

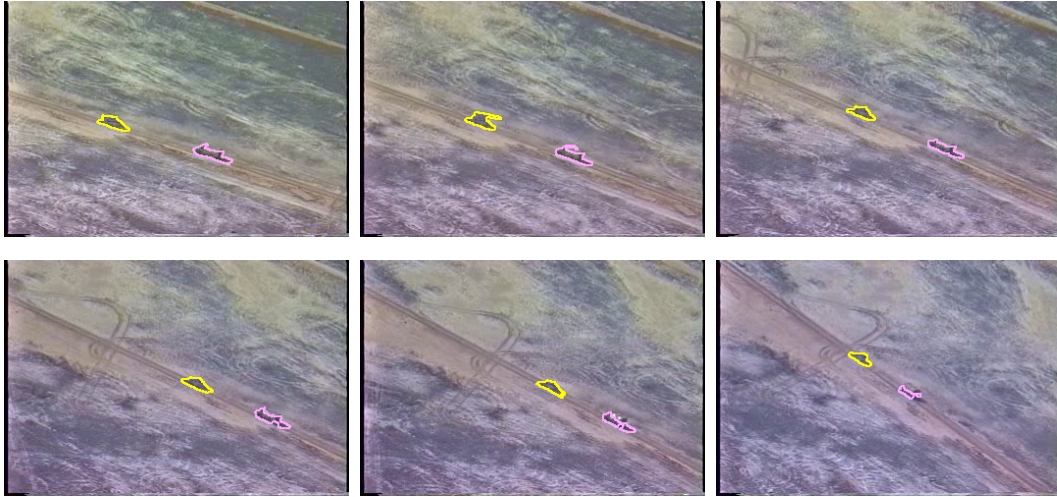


Figure 5.15: Contour tracking results “Passing EO” sequence for the VIVID dataset.



Figure 5.16: Contour tracking results “Tank EO” sequence for the VIVID dataset.

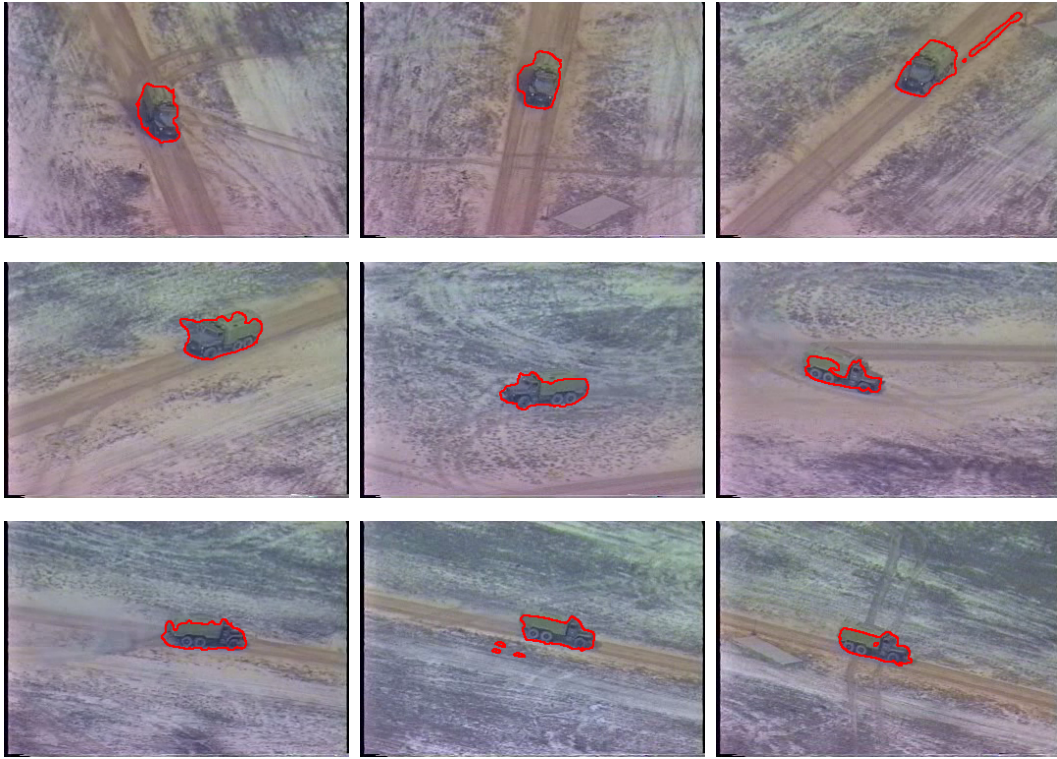


Figure 5.17: Contour tracking results “Ural EO” sequence for the VIVID dataset.

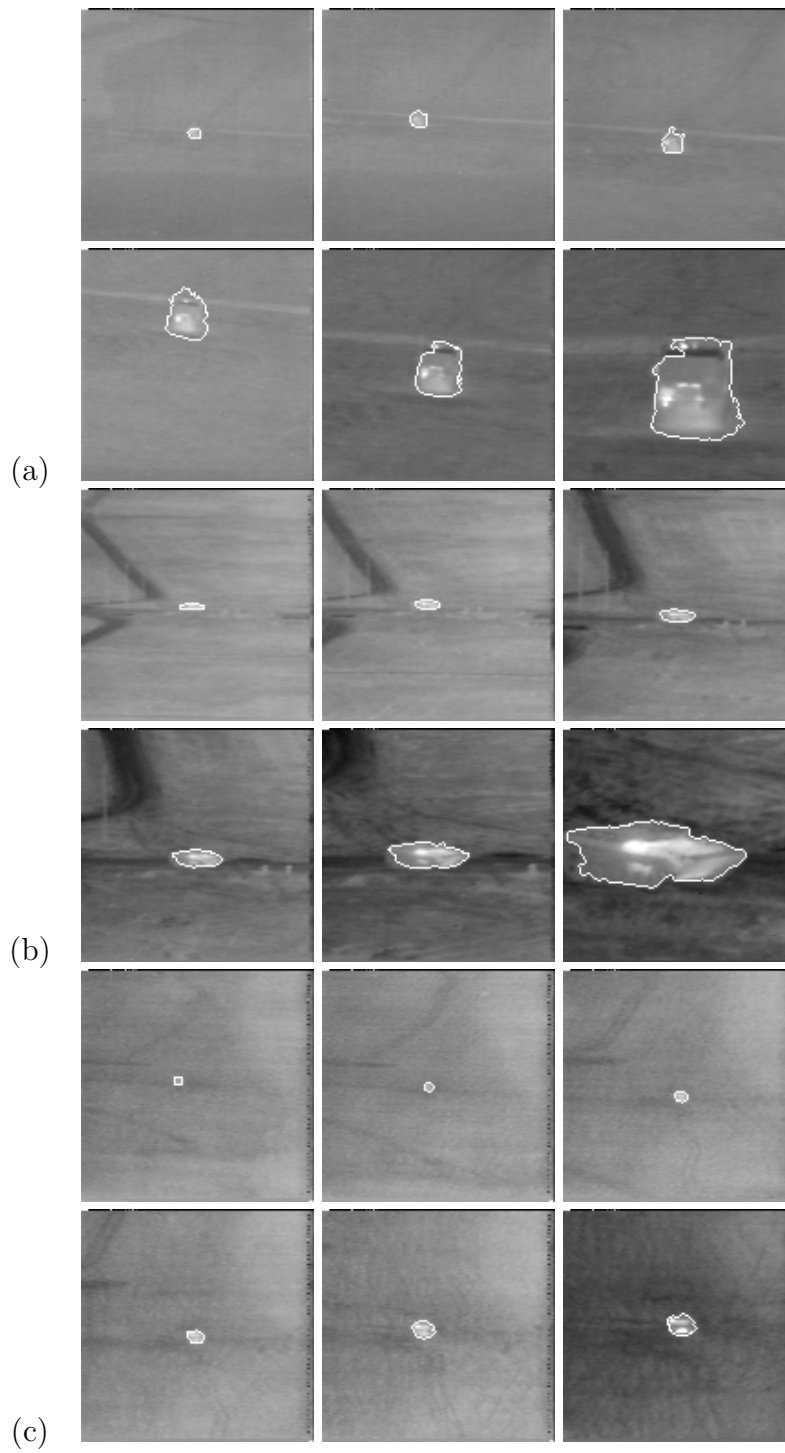
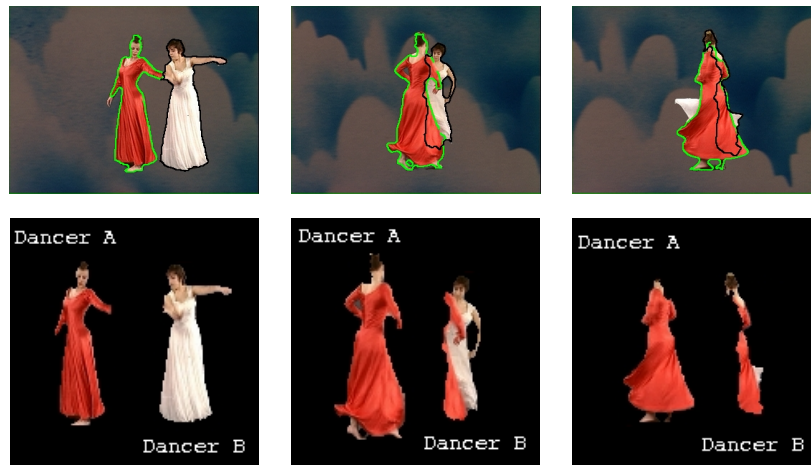


Figure 5.18: Tracking of targets in FLIR sequences taken from an airborne platform. (a) Sequence rng14_15; (b) sequence rng16_18; (c) sequence rng23_12.



(a) (b) (c)



(d) (e)

Figure 5.19: Contour tracking results for occluding dancers. The first row shows the frame from the sequence and the second row shows the extracted dancers. (a) Prior to occlusion, (b) occlusion starts, (c) full occlusion occurs, (d) dancer B is visible on the opposite side (partial occlusion), (e) occlusion ends. We recommend viewing these results in color.

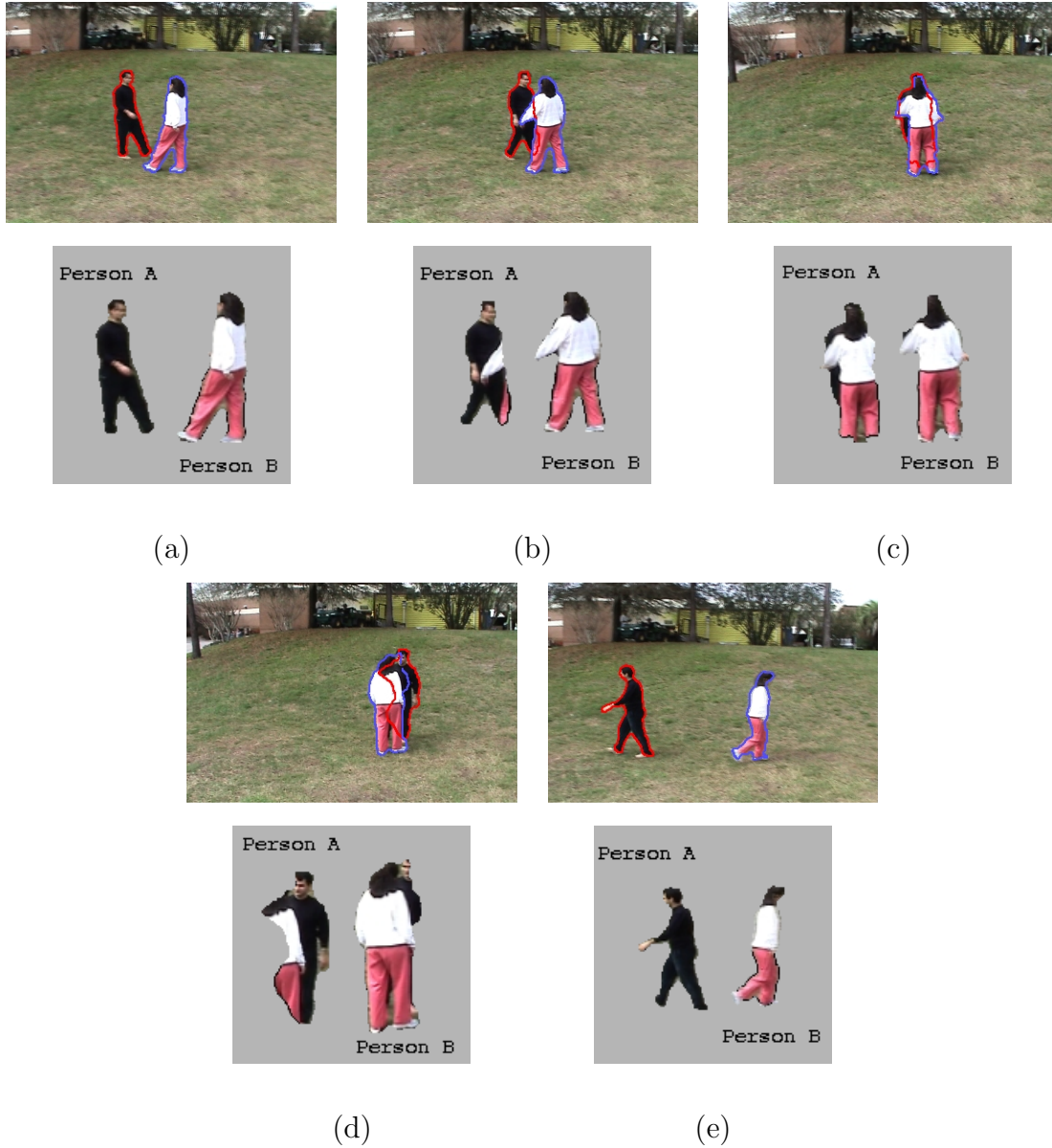


Figure 5.20: Contour tracking results for two person occlusion. The first row shows frames from the sequence and the second row shows the extracted objects. (a) Prior to occlusion, (b) occlusion starts, (c) full occlusion occurs, (d) occludee is visible on the opposite side, (e) occlusion ends. We recommend viewing these results in color.

CHAPTER 6

SPATIO-TEMPORAL VOLUME SKETCH: A NOVEL ACTION REPRESENTATION

Object tracking is usually a preprocessing step in a surveillance system. Once tracking is performed, a typical surveillance system:

- Looks for activities of persons or objects in the camera field of view,
- Generates computational models of the observed activity,
- Alerts a human supervisor when suspicious activity patterns are detected.

Activities of the tracked objects are captured by using one of the object representations described in Section 2.1. Based on the object representation chosen, there are various computational models to represent the observed activity, i.e., as trajectories of a set of tracked object features, silhouettes or 3D articulated body models. Building computational models, however, is not an easy task due to the loss of one dimension when the 3D world is projected onto a 2D image.

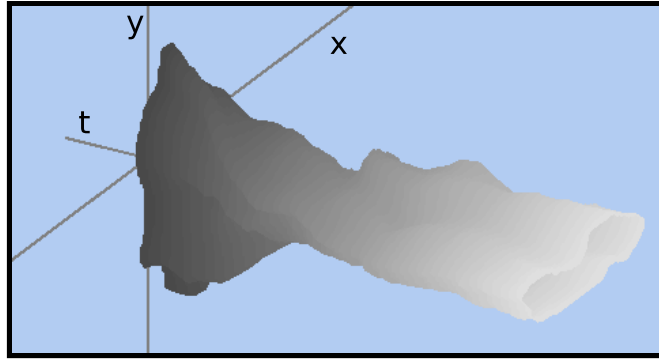


Figure 6.1: 3D action volume of a falling person.

In general, video contains lots of data, most of which is redundant. Therefore, to develop robust recognition and retrieval methods, the first step is to extract useful features from the raw video. In recent years, there has been a significant amount of work on this topic, which can be broadly classified into object-based [BD01, YRS02, SP96, WWJ03, MTH03, YSS02] and image-based approaches [LL03, BY97, HZ99, DD03, SBL02, YS00]. The object-based approaches analyze the object properties, such as the speed, shape, and motion of the object. Alternatively, image-based methods analyze the content of the complete image, such as color, texture, temporal gradients or optical flow of each image pixel. For a detailed information on features used for content based image retrieval, we refer the reader to the survey by Huang et al. [HRC99]. The object-based approach proposed by Bobick and Davis [BD01] models the changes of the silhouette by using temporal templates for representing the motion of human actors in video. Their approach does not use the motion of the object explicitly, rather it analyzes the motion history from a set of silhouettes extracted while the actor is performing the action. In [YRS02], Rao et al. use the maxima of the spatio-temporal

curvature of a trajectory generated from a single point on the object. This method is limited to the human actions that can be represented by a single trajectory, such as picking up an object. In [SP96], Starner and Pentland use HMMs to model the consecutive states of the hand for recognizing American sign language. Similar to [YRS02], Wu et al. analyze the trajectories of objects in surveillance video to recognize suspicious activities. Laptev and Lindeberg [LL03] analyze the image content to find temporal changes in the image intensities assuming that the changes are due to only the object motion. Note that intensity changes that do not belong to the object signify false action characteristics. In [HZ99], Hanjalic and Zhang propose a video retrieval approach based on extracting the representative frames (key frames) from video similar to [YS00]. Although global image features derived from key frames are the most common features used for representing the video content, the relation between the human perception and the representation of the video content using key frames is not clear. Black and Yacoob [BY97] study the human perception of facial expressions which employs the motion of facial features. They propose an automatic method for facial expression recognition using affine motion parameters of different parts of the face like the eyes and the mouth.

In this chapter, we propose a novel approach to model the motion and the shape of the moving objects for recognizing and retrieving actions. The proposed method models the variations of these features both in space and time. When an actor moves in 3D, points on the object generate tracks in four dimensions (x, y, z, t) . Projection of the 4D tracks to spatio-temporal space results in 3D trajectories. Instead of using a single trajectory,

we use all the trajectories along with their spatial relations. Spatio-temporal trajectories construct a volume, which we call the “action volume” (see Figure 6.1). This volume forms a 3D object in the spatio-temporal space. Important events, such as changes in motion and shape, represent the characteristics of the action, and are directly related to the differential geometric properties of the action volume, such as peaks, pits, valleys and ridges. These properties are invariant to viewing direction. The action volume has several advantages:

- It captures both spatial and temporal information of an action in one unified manner.
- It is a continuous representation and two sequences of the same action with different lengths will generate the same action volume. Therefore, time warping during retrieval is not necessary.
- Since the important elements composing the action sketch are based on the convex and concave parts of object contours, which are view invariant, the action sketch is also view invariant.

The proposed representation is generated in two steps:

1. *In the first step, a set of object contours are used to generate the action volume.* Generating a volume from a set of images has been previously considered in [NA94] for walking persons. Their method fit a “manually generated” walking volume, which consists of two surfaces (right and left body parts), to a walking sequence. In their approach, volume fitting is performed using a modified superquadrics fitting approach,

which involved a large number of intricate steps, and assumed fronto-parallel motion. Here we propose to “automatically generate” the volume for any action viewed from any viewing direction. In our approach, the object contours are obtained by applying the contour tracking method proposed in [YS04] to raw video. Once the contours are available, the points on consecutive contours are associated by a dynamic programming approach, in which each association is penalized based on the proximity, alignment similarity and shape similarity.

2. *After the volume is generated, differential geometric surface properties are analyzed using the Weingarten mapping to detect important action elements (action sketch).*

We demonstrate that the action sketch is invariant to the camera view point and types of surfaces forming the action sketch are related to various types of object motions.

The organization of the chapter is as follows. In the next section, we discuss the generation of an action volume from a set of contours. Section 6.2 details how the proposed action representation is obtained and its relation to various types of object motions. In Section 6.3, we discuss both the matching of two different action sketches and the view invariance property of the proposed method. Finally, we demonstrate the performance of the proposed representation for recognizing actions performed by human actors in Section 6.4.

6.1 Action Representation

Most of the computational models represent actions by using only the temporal information, i.e., shape of the performing actor is usually discarded. However, there is a close relation between the object shape and its motion. For instance, for recognizing different actions, human observers require more than a single point trajectory or a set of feature locations that describe an action. In the following discussion, we will discuss how the action volume \mathbf{B} , which encodes the motion and shape, is generated from the video.

6.1.1 Generating the Action Volume

At this point, we assume that the contours or the silhouettes of the performing actor are available. In particular, we used the tracking method proposed in Chapter 5 to track the contours Γ^t of the object at frame t (see Figure 6.2a for tracking results of the falling action). Alternatively, one can use background subtraction [GSR98] to generate the silhouettes from which contours can easily be extracted. Contour representation provides a very simple parametric form of the complete object region. In the discrete spatio-temporal space, (x, y, t) , contours constitute a dense cloud of voxels $\mathbf{c}_i = [x_i, y_i, t_i]^T$ (Figure 6.2b). Once the voxels are obtained, the next step is to establish the correspondence between voxels in the space to generate the volume.

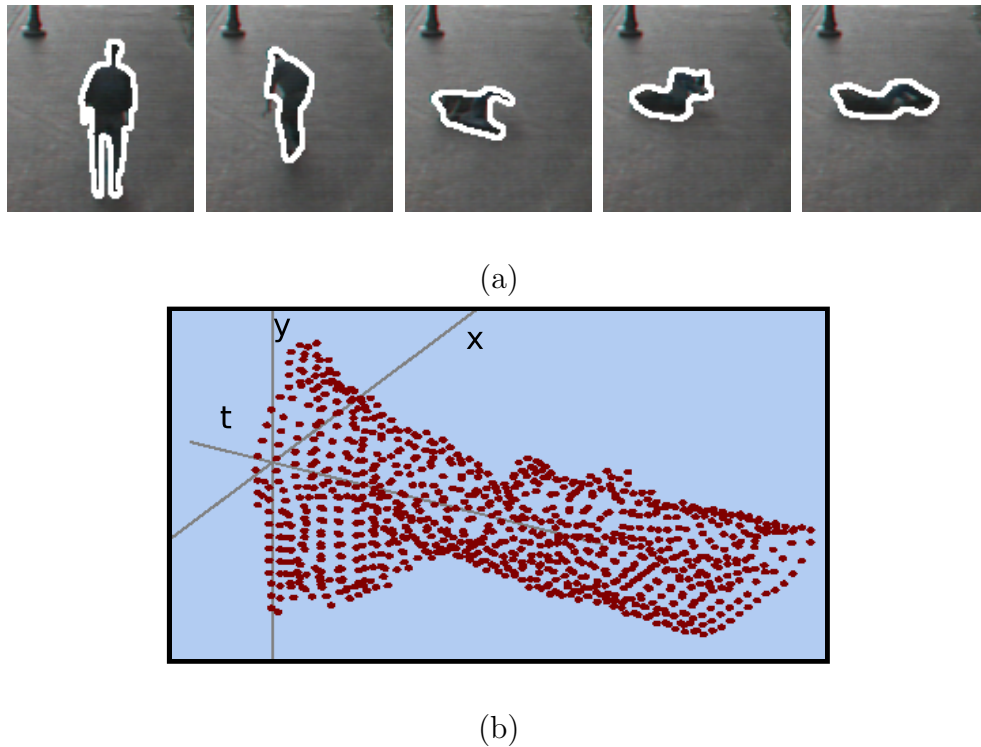


Figure 6.2: (a) A sequence of tracked object contours. (b) Clouds of voxels in the spatio-temporal space.

Matching of two point sets, either in 2D or in 3D, is still an open problem. An obvious difficulty in point matching is dealing with points that do not have corresponding points in the other set (homology). In particular, matching contours of non-rigid objects requires 1-M (one-to-many) or M-1 (many-to-one) mappings in addition to the homologies. Various methods use different constraints for point matching. An intuitive constraint is to use the nearest neighbor criterion [BM92]. Articulated rigid motion is another common constraint used by researchers [CR00]. This constraint tries to iteratively compute the transformation from a set of points in the source point cloud L (contour) to the destination point cloud R ,

assuming that a small segment of points undergo rigid motion. The set of transformations that provides the best matching while preserving the spatial neighbor criterion defines the correspondence. However, these methods require that the number of matched points in one set are equally distributed among the points in the other set, such that if one point in L is associated with 5 points in R , another point in L cannot be associated with 100 points in R .

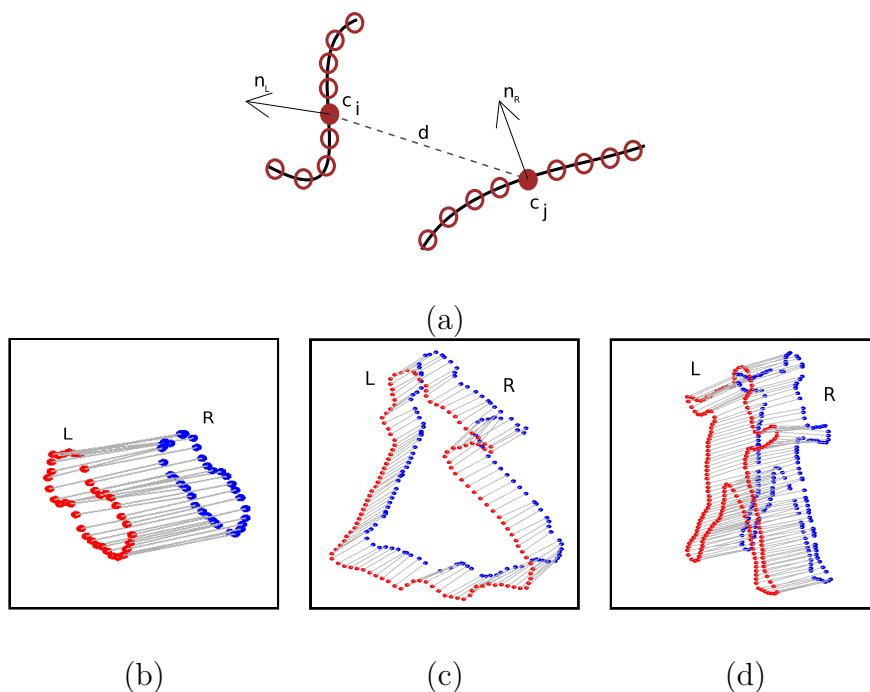


Figure 6.3: (a) Local contours from two consecutive frames used in defining the weights of central vertices. Matching of a subset of vertices between contours for (b) falling, (c) dance and (d) tennis stroke actions.

We propose to solve the correspondence problem between the contours of a non-rigid object using dynamic programming. Our approach consists of two steps.

1. The first step associates the points based on the strength of the link which is computed in terms of an association penalty. Note that at this step some points whose link strength are low remain unassociated.
2. The second step assigns the unassociated points based on the spatial arrangement of the associated points.

We define the association penalty in terms of both association from Γ^t to Γ^{t+1} and association to a particular trajectory using previous correspondences. The penalties are defined in terms of proximity, alignment and shape similarity constraints all in the spatial domain. All three constraints are computed from a set of neighboring points as in Figure 6.3a. Let $\mathbf{c}_i^k = [x_i, y_i, t]^T$ and $\mathbf{c}_j = [x_j, y_j, t + 1]^T$ be contour points in L and R respectively, where the superscript denotes the trajectory to which \mathbf{c}_i^k belongs. We compute proximity by:

$$d_{i,j} = \|\mathbf{c}_i^k - \mathbf{c}_j\|_2.$$

The alignment similarity is obtained by considering the angle α between the spatial normal vectors \vec{n}_i and \vec{n}_j corresponding to \mathbf{c}_i^k and \mathbf{c}_j respectively, which are computed in the neighborhood of the points:

$$\alpha_{i,j} = \arccos \frac{\vec{n}_i \cdot \vec{n}_j}{|\vec{n}_i| |\vec{n}_j|}.$$

Let $\mathbf{T}_{i,j} = \mathbf{c}_i^k - \mathbf{c}_j$ be the translation and

$$\mathbf{R}_{i,j} = \begin{pmatrix} \cos \alpha_{i,j} & \sin \alpha_{i,j} \\ -\sin \alpha_{i,j} & \cos \alpha_{i,j} \end{pmatrix},$$

be the rotation between the neighborhood of points \mathbf{c}_i^k and \mathbf{c}_j . Shape similarity between \mathbf{c}_i^k and \mathbf{c}_j is defined based on how the shapes of the neighborhoods N_i and N_j have changed after compensating for $\mathbf{T}_{i,j}$ and $\mathbf{R}_{i,j}$:

$$\xi_{i,j} = \sum_{\mathbf{x}_j \in N_j} \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2,$$

where $\hat{\mathbf{x}}_i = \mathbf{R}_{i,j}\mathbf{x}_i + \mathbf{T}_{i,j}$, and $\mathbf{x}_i \in N_i$ corresponds to \mathbf{x}_j based on spatial relationship (note that $|N_i| = |N_j|$). Using these three constraints, the association penalty $w_{i,j}$ from \mathbf{c}_i^k to \mathbf{c}_j is given by:

$$w_{i,j} = \exp\left(-\frac{d_{i,j}^2}{\sigma_d^2}\right) \exp\left(-\frac{\alpha_{i,j}^2}{\sigma_\alpha^2}\right) \exp\left(-\frac{\xi_{i,j}^2}{\sigma_\xi^2}\right), \quad (6.1)$$

where σ_d controls the distance between the points, σ_α controls the alignment and σ_ξ controls the degree of shape variation. In the experiments, we chose $\sigma_d = 15$, $\sigma_\alpha = 0.5$ and $\sigma_\xi = |N_i|$. In addition, the penalty of \mathbf{c}_j being associated with the trajectory k is evaluated by computing the penalty in (6.1) for each corresponding point on the trajectory T_k , $\{\mathbf{c}_j^k : \forall \mathbf{c}_j \in T_k\}$, and \mathbf{c}_i . Combining the penalties for point association and trajectory association we obtain the iterative trajectory cost as:

$$w_i^t \leftarrow \tau w_{i,j} + (1 - \tau)w_j^{t-1} \quad (6.2)$$

where t denotes the frame number, or alternatively the row of the trajectory cost matrix, and j denotes the particular point on the trajectory (column of the trajectory matrix) at t . For each frame after correspondences are obtained, penalties related to each trajectory are stored in a trajectory cost matrix which is initially empty. After the initial set of assignments, the points which had no association due to high penalties are assigned based on the neighboring

correspondences. In Figure 6.3b, c and d, we show the final vertex matchings for three different actions.

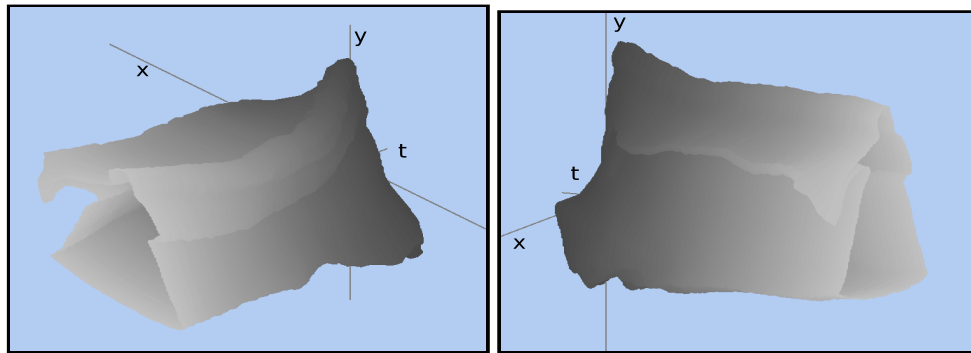
The spatio-temporal action volume can be considered as a manifold, such that it is nearly flat in small scales defined by a small neighborhood around a point. Based on this observation, a continuous volume \mathbf{B} is generated by computing plane equations in a small neighborhood around each point. In Figure 6.4a and b, spatio-temporal action volumes for tennis stroke and dance actions are shown.

Besides the implicit three dimensional (x, y, t) representation, the action volume can also be represented by a 2D parametric representation based on the arc length of the contour, s , which encodes both the object shape and the motion:

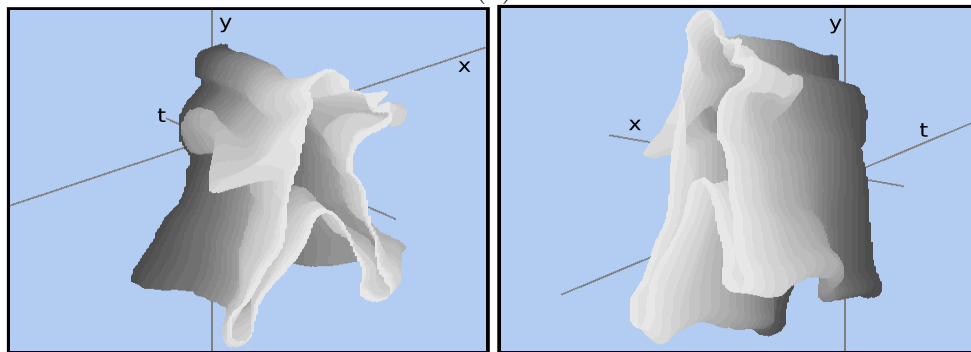
$$\mathbf{B} = f(s, t) = [x(s, t), y(s, t), t].$$

Using these two parameters, we can generate 2D motion trajectories of any point on the actor from the volume by fixing the s parameter (see Figure 6.5a). Similarly, by fixing the t parameter we can generate the object contours at time t (see Figure 6.5b).

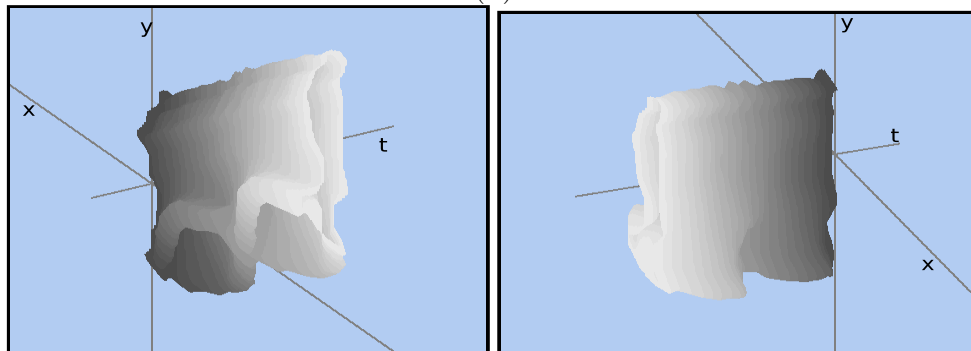
Another advantage of the action volume is that it is a continuous action representation, that is, it does not require any time warping for matching two sequences of different lengths. However, we should note that this is only valid for atomic actions. For instance, temporal invariance is valid between two walking cycles, but it is not valid between a walking cycle and a video of several walking cycles. In Figure 6.6, we show an example to demonstrate



(a)

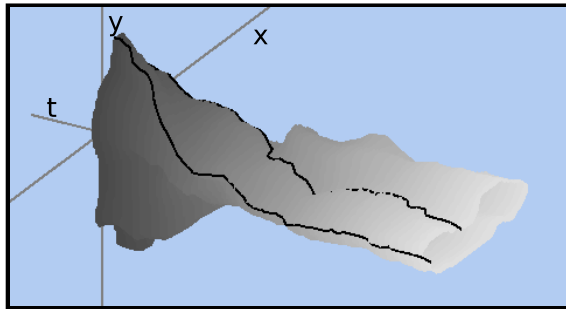


(b)

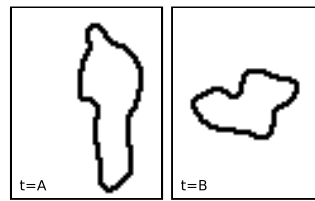
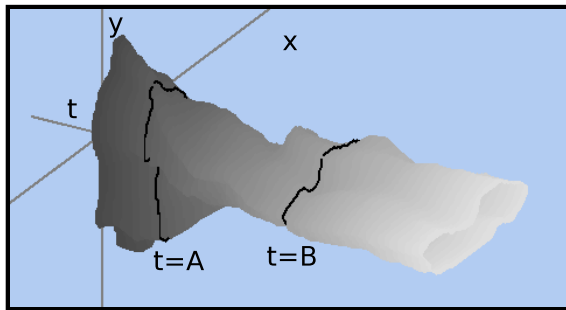


(c)

Figure 6.4: Visualization of spatio-temporal action volumes from two views for (a) dance, (b) tennis stroke, and (c) walking.



(a)



(b)

Figure 6.5: Projection of action volume. Object contours generated by fixing the (a) s and (b) t parameters in $\mathbf{B} = f(s, t)$ respectively.

the temporal invariance for the dance sequence. A new synthetic sequence with 20 frames was generated by randomly removing frames from the original dance sequence, which has 40 frames. As can be observed, the action volumes are very similar.

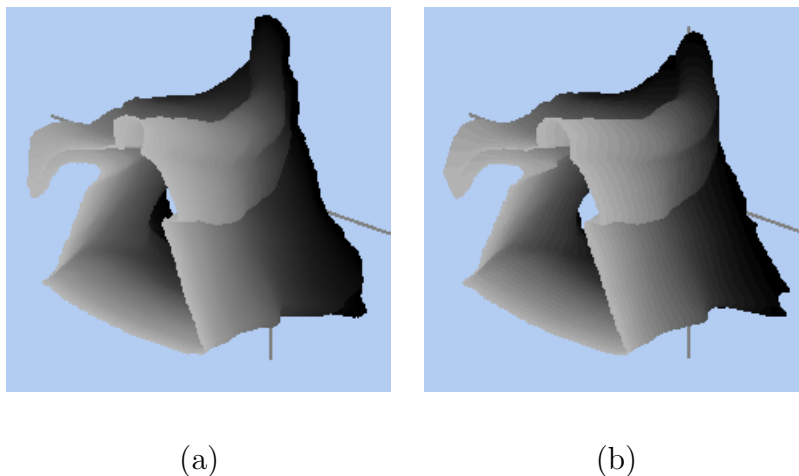


Figure 6.6: Action volumes for (a) dance sequence with 40 frames, (b) synthetic dancer sequence with 20 frames, generated by randomly removing frames.

6.2 The Action Sketch

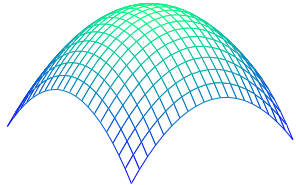
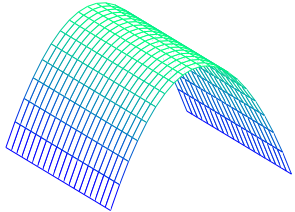
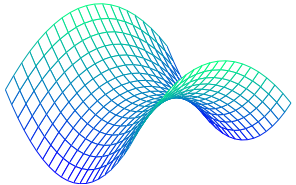
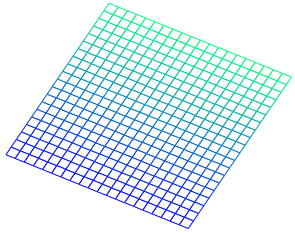
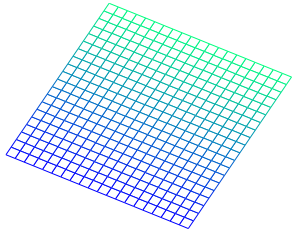
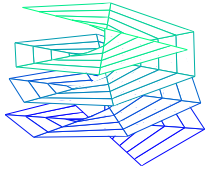
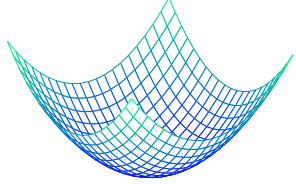
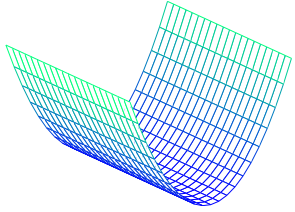
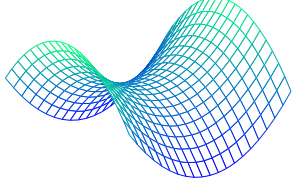
An action is typically composed of action units, which we call “action elements”. In psychology, action elements are usually identified by the important motion events, which occur due to significant changes in velocity. In our representation, every voxel in the action volume \mathbf{B} has an associated motion trajectory. Due to the spatial relationship between the points on the contour, associated trajectories are also influenced by the neighboring trajectories. These relationships and the object motion are implicitly encoded on the surface of the action volume.

To extract action elements during an action, we analyze the differential geometric properties (surface types) of the underlying volume. There are eight fundamental surface types: peak, ridge, saddle ridge, flat, minimal, pit, valley and saddle valley (see Table 6.1). Com-

pared with considering a single feature, e.g., speed or curvature, of an action trajectory, different surface types and their relation to possible object motions provides a richer representation for analyzing the video content.

Table 6.1: Fundamental surface types based on Gaussian curvature, K , and mean curvature,

H .

	$K > 0$	$K = 0$	$K < 0$
$H < 0$	peak 	ridge 	saddle ridge 
$H = 0$	none 	flat 	minimal 
$H > 0$	pit 	valley 	saddle valley 

6.2.1 Finding the Action Elements

The fundamental surface types are closely related to the object motion. For example, a peak occurs due to an instantaneous change in velocity of a convex object. These properties provide a compact description in the video domain which can be used for retrieval or recognition of action elements. As shown in Table 6.1, fundamental surface types are defined by two metric quantities, Gaussian curvature and mean curvature, computed from the Weingarten mapping which is obtained from the first and second fundamental forms of the surface [Wei02].

The first fundamental form is the inner product of the tangent vector at a given point $\mathbf{x}(s, t)$ and can be computed in the direction (s_p, t_p) by:

$$\mathbf{I}(s, t, s_p, t_p) = [s_p \quad t_p]^T \underbrace{\begin{bmatrix} E & F \\ F & G \end{bmatrix}}_{\mathbf{g}} [s_p \quad t_p], \quad (6.3)$$

where $E = \mathbf{x}_s \cdot \mathbf{x}_s$, $F = \mathbf{x}_s \cdot \mathbf{x}_t$ and $G = \mathbf{x}_t \cdot \mathbf{x}_t$, where subscripts denote the partial derivatives.

The \mathbf{g} matrix is called the metric tensor of the surface and has the same role as the speed function for spatio-temporal trajectories.

The second fundamental form defines the variation of the normal vector with respect to the surface position. It is computed by the following equation:

$$\mathbf{II}(s, t, s_p, t_p) = [s_p \quad t_p]^T \underbrace{\begin{bmatrix} L & M \\ M & N \end{bmatrix}}_{\mathbf{b}} [s_p \quad t_p], \quad (6.4)$$

where \vec{n} is unit normal vector, $L = \mathbf{x}_{ss} \cdot \vec{n}$, $M = \mathbf{x}_{st} \cdot \vec{n}$ and $N = \mathbf{x}_{tt} \cdot \vec{n}$, and subscripts denote the second order partial derivatives. In terms of encoding motion, N in (6.4) is related to the acceleration of $\mathbf{x}(s, t)$.

The Weingarten mapping combines both fundamental forms given in (6.3) and (6.4) into a single matrix S :

$$S = \mathbf{g}^{-1}\mathbf{b} = \frac{1}{EG - F^2} \begin{bmatrix} GL - FM & GM - FN \\ EM - FL & EN - FM \end{bmatrix}. \quad (6.5)$$

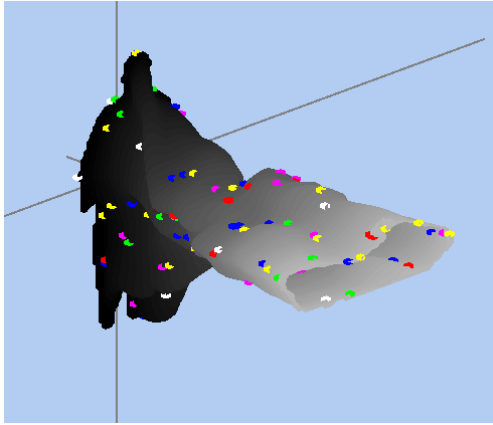
Gaussian curvature, K , and mean curvature, H , are two algebraic invariants derived from Weingarten mapping [BJ86]. In particular, Gaussian curvature is the determinant of S :

$$K = \det(S) = \frac{LN - M^2}{EG - F^2},$$

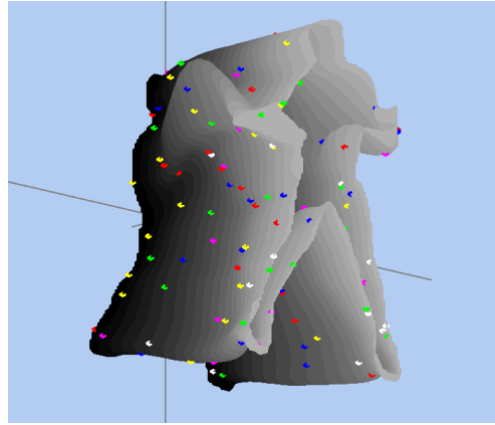
and the mean curvature is half of the trace of S :

$$H = \frac{1}{2}\text{trace}(S) = \frac{EN + GL + 2FM}{2(EG - F^2)}.$$

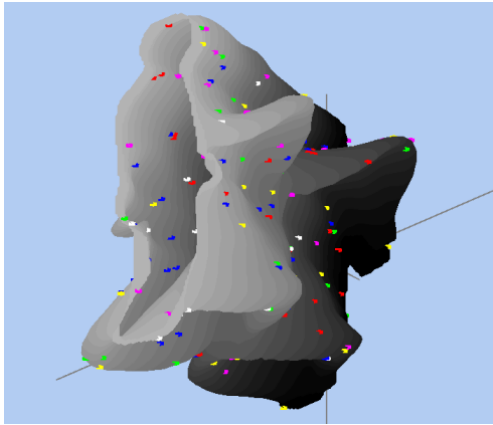
After the curvatures are computed, maxima of $|K|$ and $|H|$ in a local neighborhood indicate the presence of important events on the action volume. Note that the signs of these curvature maxima define the surface type, and thus the type of the action (see Table 6.1). We will use the curvature maxima to represent the actions. In Figure 6.7, we show several action volumes with action elements color coded according to the eight fundamental surface types.



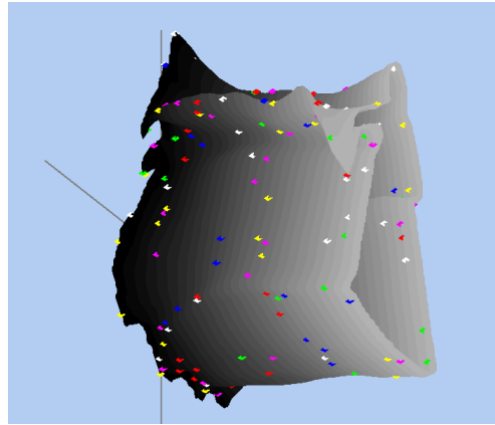
(a) Falling



(b) Tennis stroke



(c) Walking



(d) Dancing

Figure 6.7: Color coded action characteristics corresponding to various surface types. Color codes are: red (peak), yellow (ridge), white (saddle ridge), blue (pit), pink (valley) and green (saddle valley).

6.2.2 Action Elements and Their Relations to Object Motions

Action elements can be the result of shape changes, such as closing the fingers while forming a fist, or motion of a body part without changing its shape, such as a waving hand. As discussed in the previous section, the action elements are directly related to the fundamental surface types shown in Table 6.1. However, in the context of action content analysis in the video, the flat surface and the minimal surface types are not important. In particular, a the flat surface type only occurs when there is no object motion or shape change. Due to the surface structure, minimal surface does not occur in action volumes. In the rest of the paper, we will consider the remaining six surface types, namely, peak, pit, ridge, saddle ridge, valley, and saddle valley for generating the action sketch.

The analogy between the surface types and object shape and motion can be broadly given as follows:

- A flat shaped object can generate a ridge, a valley or a flat surface.
- A convex shaped object can generate a peak, a ridge or a saddle ridge surface.
- A concave shape object can generate a pit, a valley or a saddle valley surface.

To simplify the discussion, we will concentrate on simple wire object moving on a planar surface. Possible object motions for the wire object include: no motion, constant speed in one direction, and deceleration followed by an acceleration in the opposite direction (includes a full stop).

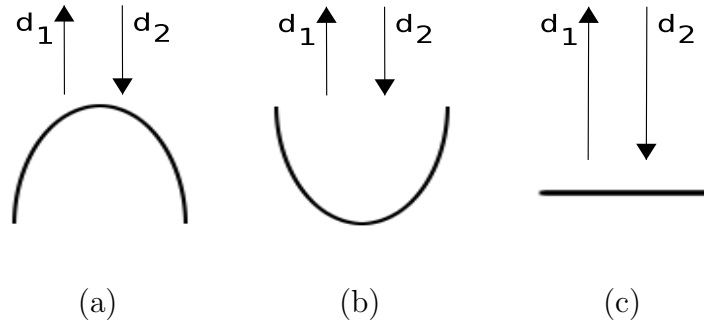


Figure 6.8: Directions of motion for (a) convex wire, (b) concave wire, (c) straight wire.

- **Peak:** This surface type can only be generated by the convex shaped wire object (Figure 6.8a) moving first in direction d_1 as in Figure 6.8a, then in the direction of d_2 . A peak usually occurs when the object moves suddenly. A peak is a common action characteristic which is distinctive for different actions.
- **Pit:** This surface defines characteristics similar to the peak surface, but it is defined for the concave wire object as shown in Figure 6.8b. The motion first should be in the d_2 direction then in the d_1 direction. A pit occurs during sudden object motion, and is a common action characteristic.
- **Ridge:** A ridge occurs in two different ways. The first possibility is that the concave wire object given in Figure 6.8a moves with a constant speed in any direction, or does not move at all. The second possibility is the straight wire shown in Figure 6.8c moving first in the d_1 direction then in the d_2 direction. Although a ridge can occur when there is no motion, since it also occurs during object motion, we consider this surface type an important action characteristic.

- **Saddle Ridge:** A saddle ridge is defined for the convex object (Figure 6.8a) and can only be created by object motion. The object should move in opposite directions for peak generation, i.e., first in the d_2 direction then in the d_1 direction.

The discussion for the ridge and saddle ridge can simply be extended to the valley and the saddle valley. The difference between ridges and valleys is that the local object contour has to be concave for the latter. Object concavity or convexity and the object motion is encoded by the values of the mean and the Gaussian curvatures. For the peak and the pit surface types the mean curvature encodes the shape of the object (concave: $H < 0$, convex $H > 0$) and the Gaussian curvature shows the bending of the temporal surface in the time t direction, such that when $K > 0$, the object moves in the normal direction of the object contour and when $K < 0$ it moves in the direction opposite to the contour normal. Similar arguments hold for the saddle valley and saddle ridge surfaces. However, for the valley and the ridge surfaces, object shape and motion can be encoded by both the mean and the Gaussian curvature. Depending on the surface type, the curvatures can also show the motion direction and speed of a voxel.

6.3 View Invariant Action Recognition

Volumes representing the actions can be considered three-dimensional rigid objects, and the action sketch contains the representative features of this 3D (x, y, t) object. In this setup,

both the retrieval and the recognition problems become matching 3D object features with the features of other objects of known types. In this chapter, we propose a general graph theoretical framework to match action elements obtained from different viewpoints. We will first discuss the view invariance of the action sketch, then we will discuss the matching strategy to compute a matching score between two different actions.

6.3.1 View Invariance

One of the fundamental problems in video retrieval is to define the features that are invariant to the viewing direction. There are unfortunately very few approaches that claim view invariance for action recognition. In [YRS02], the authors used the maxima of the spatio-temporal trajectory as the view-invariant features under an affine camera model. Seitz and Dyer [SD97] proposed an affine based view-invariant method to find repeating poses of walking people. Here, we will present the view invariance properties of the proposed action sketch.

As discussed in Section 6.1.1, volume \mathbf{B} is generated by computing the correspondence between points on the object contours Γ^t over a temporal period. Thus, if the contours are view invariant, then action elements will also be view invariant. In particular, the elements are defined when mean curvature $H > 0$ or $H < 0$. This only happens at surface patches, that correspond to concave or convex parts of object contours. Here, we are interested in

showing that the concavity or convexity of the contour is independent of the camera view point.

Let's first consider a concave segment in a contour, which is defined by two components: the normal vector $\vec{\mathbf{n}}$ and the angle α defined by $\mathbf{P}_1 = [X_1, Y_1, Z]$, $\mathbf{P}_2 = [X_2, Y_2, Z]$ and $\mathbf{P}_3 = [X_3, Y_3, Z]$ as shown in Figure 6.9a. We claim that from any viewpoint in 3D space, $0^\circ < \alpha < 180^\circ$ always holds for an affine camera.

Let the transformation of the view point in 3D be defined by the rotation matrix R and translation vector T . Since T does not affect α , we will drop T . In addition, rotation around the z-axis also does not affect α . Thus, we are left with only R , which has two components the pan angle β (rotation around the y-axis) and the tilt angle γ (rotation around the x-axis). When the camera pans, the coordinates of X change as follows:

$$X' = X \cos \beta - Z \sin \beta,$$

$$Y' = X \sin \beta + Z \cos \beta.$$

Before projecting from 3D into the image plane, we will first translate and rotate the contour segment such that \mathbf{P}_2 is the origin and the normal $\vec{\mathbf{n}}$ points in the y direction: $\tilde{\mathbf{P}}_1 = R_\theta(\mathbf{P}_1 - \mathbf{P}_2)$, $\tilde{\mathbf{P}}_2 = [0, 0, 0]$ and $\tilde{\mathbf{P}}_3 = R_\theta(\mathbf{P}_3 - \mathbf{P}_2)$, where $\theta = \arccos \frac{\vec{\mathbf{n}}_y}{\|\vec{\mathbf{n}}\|}$. Due to this transformation, we have the following (see Figure 6.9b):

$$\tilde{X}_1 < 0, \tilde{Y}_1 < 0, \tilde{Z}_1 = 0, \tilde{X}_3 > 0, \tilde{Y}_3 < 0, \tilde{Z}_3 = 0. \quad (6.6)$$

Image coordinates under an affine camera model are given by $x = f \frac{X}{D}$ and $y = f \frac{Y}{D}$, where D is the distance from the camera and f is the focal length. The projection of $\tilde{\mathbf{P}}_i : i = 1, 2, 3$

to \tilde{p}_i on the image plane is

$$\tilde{p}_i = \left[f \frac{\tilde{X}_i \cos \beta}{D} \quad f \frac{\tilde{X}_i \sin \beta}{D} \right]^T \quad (6.7)$$

for $i = 1, 3$ and $\tilde{p}_2 = [0, 0]^T$. Our constraint for concavity $0^\circ < \alpha < 180^\circ$ implies that the x and y components in (6.7) have to follow the conditions in (6.6). Dropping the constants f and D , we get $\tilde{X}_1 \cos \beta < 0$, $\tilde{X}_1 \sin \beta < 0$ and $\tilde{X}_3 \cos \beta > 0$. These constraints hold when $-90^\circ < \beta < 90^\circ$, which means that concavity is maintained when the camera pans within a hemisphere. The same argument holds when the camera tilts around the x-axis, such that $-90^\circ < \gamma < 90^\circ$. In addition, the camera can pan and tilt simultaneously in the hemisphere defined by β and γ . Since the convex and concave segments in a contour are view invariant, the spatio-temporal action sketch will also remain view invariant.

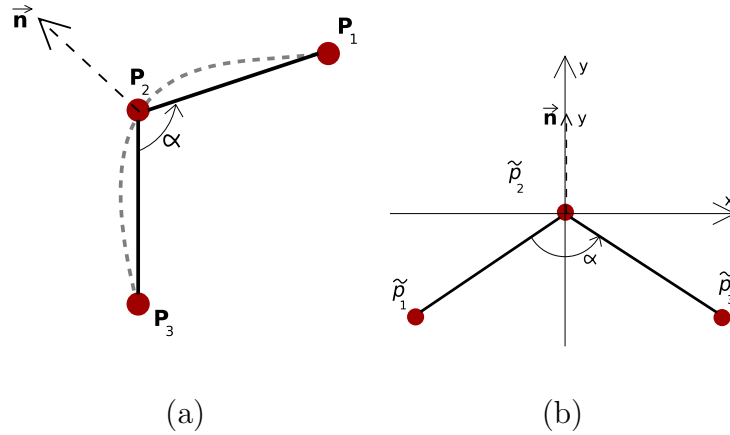


Figure 6.9: (a) Concave contour segment denoted by three control points and (b) a normal rotated concave contour segment.

6.3.2 Matching Actions

Let \mathbf{D}_i be the i^{th} action volume in the action database, and \mathbf{B} be the input action volume. For each action volume, let the action elements be represented by $V_\alpha(s, t)$, where $\alpha = \{\mathbf{B}, \mathbf{D}_i\}$. Let G be a bipartite graph where the vertices are the elements $V_\alpha(s, t)$. The edges are defined from $V_{\mathbf{B}}(s, t)$ to $V_{\mathbf{D}_i}(s, t)$, and there are no edges between the vertices within an action volume.

The weights of the edges between the nodes have two components. The first component measures the proximity in the spatio-temporal domain and the second component measures the similarity of the surface shape. The proximity between the k^{th} node in \mathbf{B} and the l^{th} node in \mathbf{D}_i is defined by the Euclidean norm in three dimensions:

$$d_i(k, l) = \| V_{\mathbf{B}}(s_k, t_k) - V_{\mathbf{D}_i}(s_l, t_l) \|_2 .$$

The similarity of the local surface defined in the neighborhood of a node measures the degree of shape variation and object motion based on the Gaussian curvature K and mean curvature H :

$$\Psi_i(k, l) = \tau e^{-\frac{(K^k - K^l)^2}{\sigma_K^2}} + (1 - \tau) e^{-\frac{(H^k - H^l)^2}{\sigma_H^2}} ,$$

where τ controls the importance of each component. The weight of the edge is then given by convex combination of the proximity and shape similarity features:

$$w_i(k, l) = .5 \exp\left(-\frac{d_i^2(k, l)}{\sigma_d^2}\right) + .5 \Psi_i(k, l)$$

In our experiments, we have chosen $\sigma_K = 10$, $\sigma_H = 10$ and $\tau = 0.6$. The correspondence is then achieved by computing the maximum matching of the bi-partite action graph. In Figure 6.10, we show matchings between the tennis stroke and the falling actions. In the figure, unassociated nodes correspond to associations with extremely low weights.

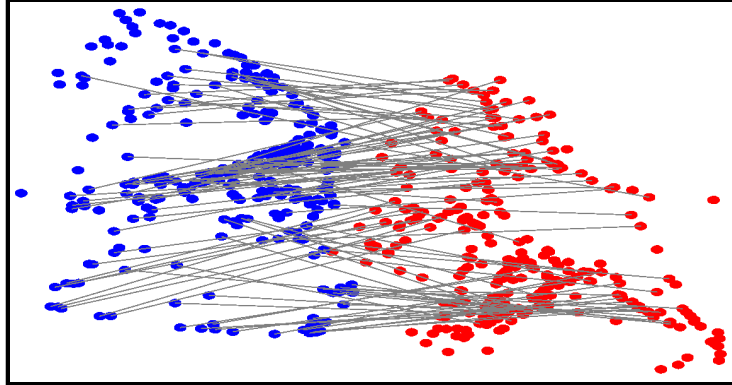


Figure 6.10: Associated vertices using maximum graph matching. The red vertices are from the tennis stroke action, the blue vertices are from the falling action. Some vertices have no correspondence due to 0 weight.

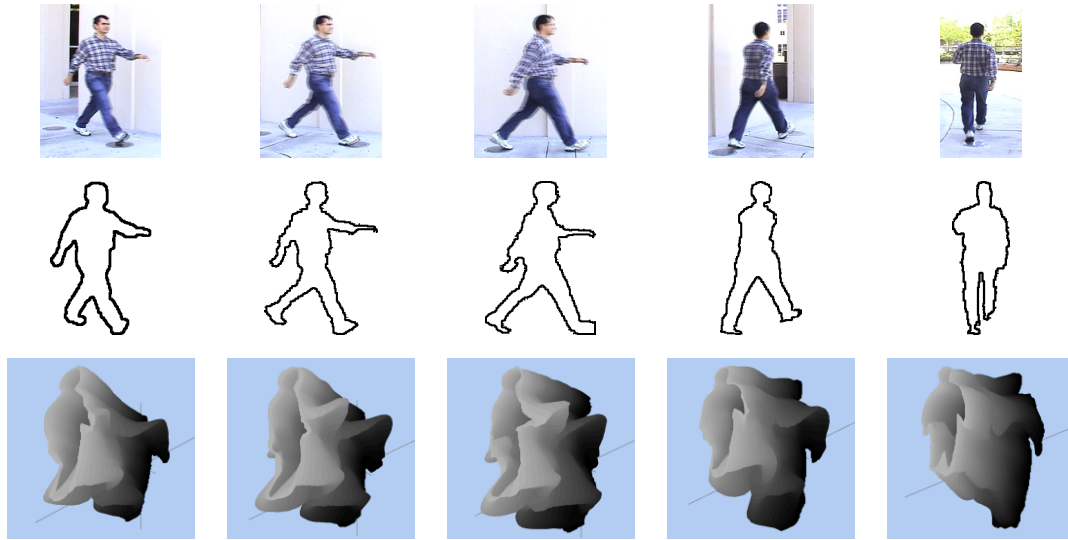
Once the corresponding elements $V_{\mathbf{B}} = [s, t]^T$ and $V_{\mathbf{D}_i} = [s', t']^T$ in two actions are determined, the fundamental matrix between the elements generated from two uncalibrated cameras is defined by: $V_{\mathbf{B}}^T \mathcal{F} V_{\mathbf{D}_i}$. Using several such correspondences between two action

sketches we have

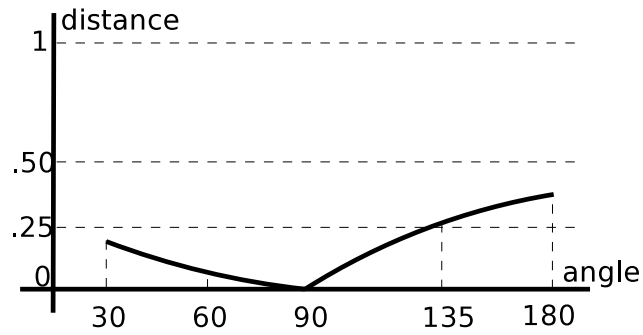
$$\mathcal{A}f = \begin{bmatrix} s'_1 s_1 & s'_1 t_1 & s'_1 & t'_1 s_1 & t'_1 t_1 & t'_1 & s_1 & t_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s'_n s_n & s'_n t_n & s'_n & t'_n s_n & t'_n t_n & t'_n & s_n & t_n & 1 \end{bmatrix} f = 0$$

where f is a 9-element vector (in row-major order) representation of \mathcal{F} . Since the above equation is a homogeneous equation, \mathcal{A} has a rank of at most eight, if the two sketches match. The similarity of two actions can then be measured using $\rho_{\mathbf{B}, \mathbf{D}_i} = \sigma_9^2/N$, where σ_9^2 is the ninth singular value of \mathcal{A}^2 . After the distances from the input action to all actions in the database are computed, recognition is achieved by selecting the action from the database that has the minimum distance, $\arg \min_i(\rho_{\mathbf{B}, \mathbf{D}_i})$. This matching is invariant to viewing direction.

In Figure 6.11a-e, we show the walking action captured from different viewpoints and their corresponding action surfaces. In Figure 6.11f, the distances of the fronto-parallel walking (Figure 6.11c) from the other views is plotted as a function of viewing angle. As can be observed in Figure 6.11f, the distances on the left side of 90° are lower than the distances on the right side of 90° . This is mainly due to the self occlusion of the right arm in the videos captured from viewing angle greater than 90° . In general, even though some body parts are not available in the action volume due to self occlusion, the visible parts provide adequate information as evident from low distances in Figure 6.11f.



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



(f)

Figure 6.11: Walking action from five different view points, first row shows one frame of the action, second row shows associated contours and third row shows the corresponding action volumes. (a) 30° , (b) 60° , (c) 90° , (d) 145° , (e) 180° . (f) Matching scores, for the action in (c) with the other actions.

6.4 Experiments

In order to test the performance of the proposed approach, we use twenty-eight sequences of twelve different actions captured from different viewing angles. The video sequences include dancing (2), fall (1), tennis strokes (2), walk (7), run (1), kick (2), sit down (2), stand up (3), surrender (2), hands-down (2) and aerobics (4) actions (the number in the parenthesis denote the count of actions). In Figures 6.12 and 6.13, we show the complete set of action sketches superimposed on the action volumes.

From input video footage, we first track the contours of the objects and generate the action volume as detailed in Sec. 6.1.1. For each action, the action sketch is generated by analyzing the differential geometric surface properties of the underlying volume. The action sketch is then matched with the representative actions in the database by computing the view invariant distance measure discussed in Sec. 6.3. In Table 6.2, we tabulate the two best matches of each action video. For all but five actions the best matches of action videos are correct. For the remaining five the second best matches are correct. Usually, the matching criterion clusters similar actions during retrieval. Specifically for the second video, in which the action is recognized as the second best match, “hands down” action involves “standing up” and the proposed method retrieved both actions. The hands down action also shows the importance of using both the shape and the motion of the object. For instance, if we were modeling the trajectory of only one hand, we would end-up with a line shaped trajectory

which is not a characteristic of the action. However, the shape variation around the knees (due to standing up) and the motion of the hands identifies the action.

6.5 Conclusions

We propose a novel approach to represent and retrieve actions using action sketches which are generated from spatio-temporal action volumes. Given the object contours for each time slice, we first generate an action volume by computing point correspondence between consecutive contours using dynamic programming. This volume can be treated like an object in the spatio-temporal space. The action volume consists of spatio-temporal trajectories for voxel. In order to obtain a compact action representation, we analyze differential geometric surface properties of the underlying volume. These properties, which we call the elements of the “action sketch”, are related to the object motion and the shape through the surface invariants, namely the Gaussian and the mean curvatures. The action sketch defines important elements during an action and is invariant to the viewing direction, such that the action elements of the sketch occur at the same positions regardless of the camera position. Finally, using these view invariant features, we retrieve videos of similar action content based on a view-invariant distance measure. Experiments performed on twenty-eight videos show promising results.

Table 6.2: Recognition results for various actions. Bold face represents the correct matches and italics indicate the best match is incorrect.

Video	First best	Second best	Video	First best	Second best
1	Dance	Walking	15	Walking	Kicking
2	<i>Stand up</i>	Hand down	16	<i>Hands down</i>	Stand up
3	Walking	Kicking	17	Surrender	Aerobic 1
4	Kicking	Stroke	18	Tennis stroke	Walking
5	Walking	Kicking	19	<i>Aerobic 4</i>	Walking
6	Stand up	Hands down	20	Dance	Aerobic 3
7	<i>Hands down</i>	Surrender	21	Aerobic 2	Aerobic 3
8	Hands down	Kicking	22	Aerobic 3	Kicking
9	Kicking	Aerobic 1	23	Sit down	Dance
10	Falling	Kicking	24	Walking	Kicking
11	Walking	Kicking	25	Aerobic 4	Dance
12	<i>Sit down</i>	Walking	26	Tennis stroke	Kicking
13	Aerobic 1	Walking	27	Stand up	Release
14	Sit down	Falling	28	Running	Kicking

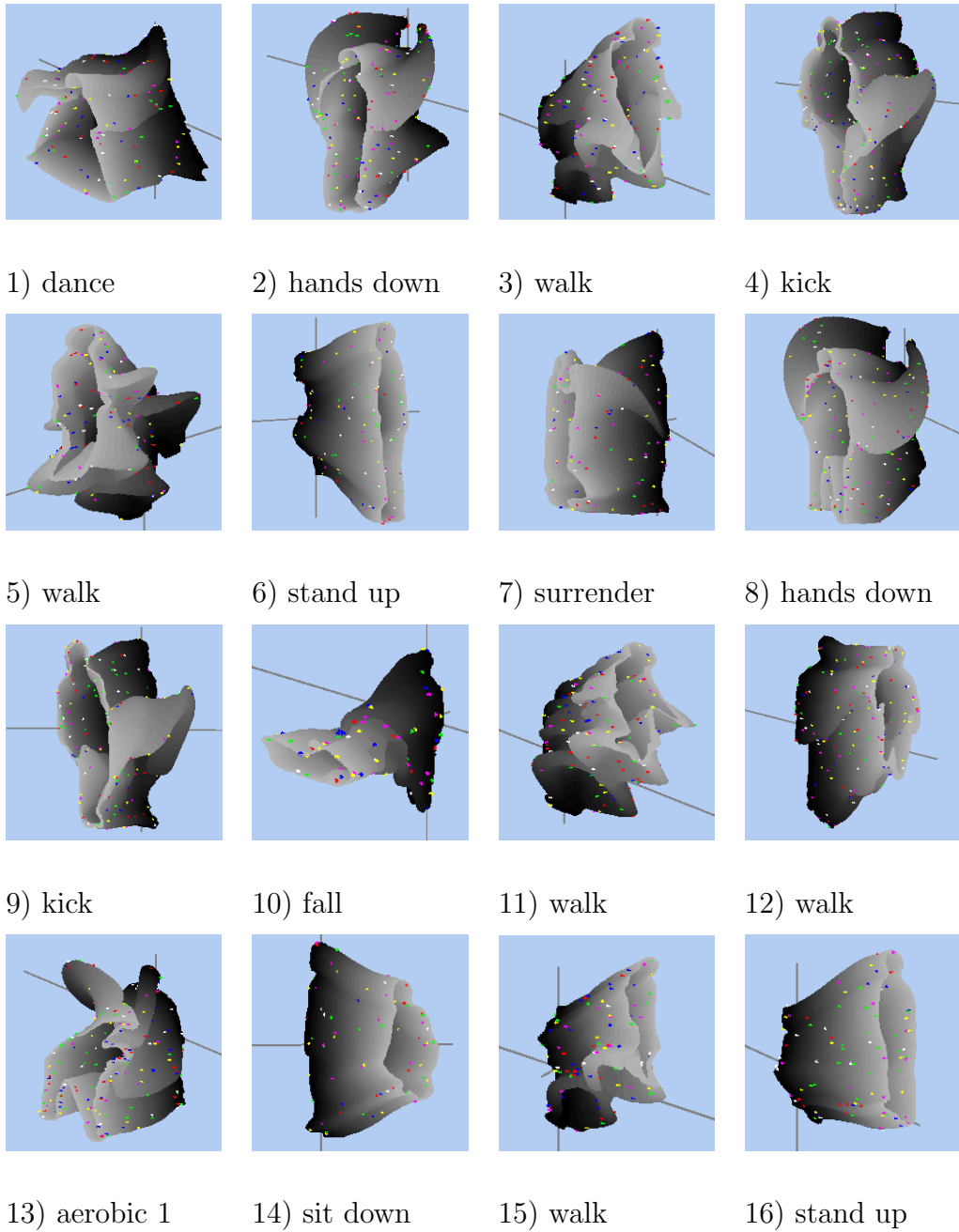
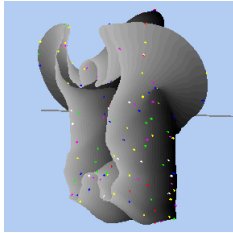
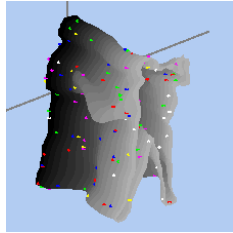


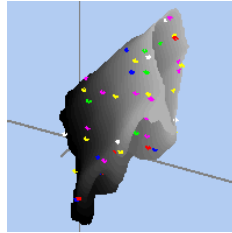
Figure 6.12: Action sketches superimposed on the action volumes, which are generated from video sequences of human actors. Numbers denote the action labels in Table 6.2. Continues in Figure 6.13.



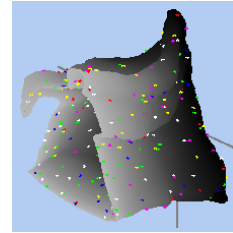
17) surrender



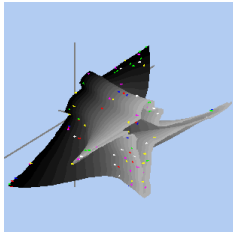
18) stroke



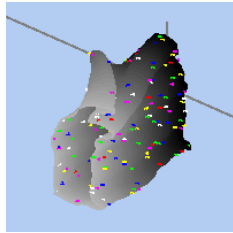
19) walk



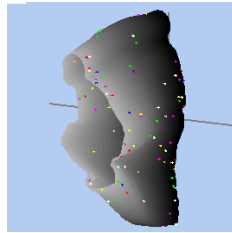
20) dance



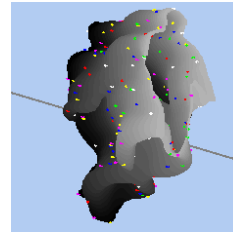
21) aerobic 2



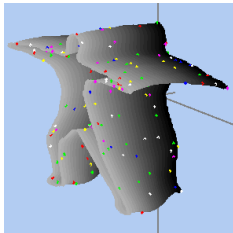
22) aerobic 3



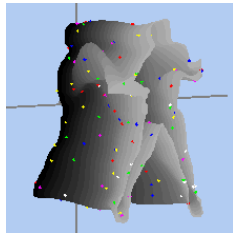
23) sit down



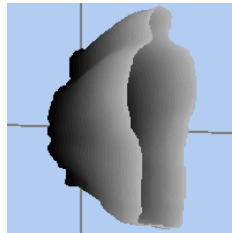
24) walk



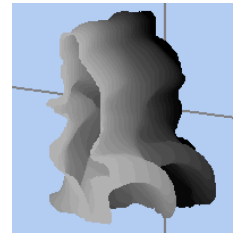
25) aerobic 4



26) stroke



27) stand up



28) run

Figure 6.13: Continuation of Figure 6.12.

CHAPTER 7

CONCLUSIONS AND FUTURE DIRECTIONS

The proliferation of high powered computers increased demand of surveillance systems, which in turn rocketed the number of papers on tracking and surveillance in the last decade. Despite the popularity, there has been only a handful of surveys on only specific topics within the area of tracking, however no single survey on tracking in general. In Chapter 2, we tried to address this need and provided a comprehensive survey on the state-of-the-art. The survey:

- follows a step by step discussion on how to propose a new tracking method from selecting the object representation to selection of the tracking methodology,
- discusses object detection mechanisms, which is required for all the tracking methods,
- provides an intuitive classification of different tracking approaches,
- proposes possible evaluation criteria for different classes of tracking methods,
- discusses challenging tasks and future directions.

With this survey, we hope to identify new trends and ideas to solve the tracking problem and introduce the researchers to tracking field.

Following the classification provided in Chapter 2, we proposed two tracking methods in Chapters 4 and 5. Both of the methods perform tracking for video acquired using mobile cameras, where the object models and the background models change overtime:

- The first tracking method is a patch based. It uses color and local intensity deviation to represent the objects during tracking. For each object detected in the first frame, kernel density estimate of the color and local intensity deviation is computed. Tracking is then performed by first initializing the object position with its previous spatial position, then computing the shift vector which maximizes the likelihood of observing the object given the a priori object model. The maximization is an iterative procedure and usually it takes up to six iterations to locate the object. Proposed object tracker also uses dynamic models and updates them automatically when it observes changes in the color and local intensity deviation features. Through global motion compensation it can track objects for aerial videos, where sensor motion generates abrupt object motion. Global motion is compensated when tracker fails to locate the object in the current frame.
- The second tracking method tracks the contours of the objects detected in the first frame. Proposed contour tracker uses color and texture to model the objects and background. In addition, conditioning the tracking of the object contour on previous contours results in shape priors which increase the robustness of tracking during partial or full occlusions. The proposed tracking functional, which uses a support region around the contour, suppresses the noise and artifacts that generally occur during the

course of tracking. The minimization of the proposed tracking energy functional is performed by evolving the contour in the steepest descent direction and the contour is implicitly represented using level sets. During the course of tracking, if there is no occlusion that distorts the shape, the evolution speed in the level set representation is based purely on the visual features such as color and texture. When occlusion happens, the proposed occlusion detection and handling method successfully estimates the contour of the occluded regions using a level set evolution speed based on the shape prior learned over time. The results presented show the robustness of the tracking algorithm with and without occlusion for both EO and IR imagery captured from moving cameras.

Tight object contours obtained from the object tracker proposed in Chapter 5 are suitable for higher level tasks such as behavior analysis. Following this observation, in Chapter 6, we propose a novel approach to represent and retrieve actions using action sketches which are generated from spatio-temporal action volumes. Given the object contours for each time slice, we first generate an action volume by computing point correspondence between consecutive contours using dynamic programming. This volume can be treated like an object in the spatio-temporal space. The action volume consists of spatio-temporal trajectories for voxel. In order to obtain a compact action representation, we analyze differential geometric surface properties of the underlying volume. These properties, which we call the elements of the “action sketch”, are related to the object motion and the shape through the surface invariants namely the Gaussian and the mean curvatures. Action sketch defines important

elements during an action and is invariant to the viewing point, such that the action elements of the sketch occur at the same positions regardless of the camera position. Using these view invariant features, we propose a view invariant action recognition methods based on the rank of the fundamental matrix computed from the spatio-temporal positions of the action sketch.

7.1 Future Directions

The use of a particular feature can greatly affect the tracking performance. Generally the features that best discriminate between the object and the background are also best for tracking purposes. However, the same feature may not be the most discriminative one throughout a sequence. The tracking methods proposed in this thesis use a weighted combination of color and texture features where the weights are automatically computed. However, color and texture are not necessarily the best features for an object, and there are other features that may provide unique object signatures. One possible direction is to consider different features such as motion and different filter responses. Then, fusion of the features can be performed by Adaboost to obtain the best tracking performance. Adaboost is method of finding a strong classifier based on a combination of moderately inaccurate “weak” classifiers. Given a large set of features, one classifier can be trained for each feature. Adaboost will discover weighted combination of classifiers (representing features) that maximize the classification performance of the algorithm. The higher the weight of the feature the more discriminatory it is.

Compared to computational cost of the tracking method proposed in Chapter 4, contour tracking method proposed in Chapter 5 is not real time. One of the reasons for high computation cost is the requirement of iterative computations per level set grid. This can be reduced dramatically by switching to an explicit contour representation, e.g., splines with control points, in stead of the implicit level set representation. Another possibility is to propose a modified level set scheme, similar to the narrow band approach for shrinking contours [Set99], which does not require evaluation of each grid in the level set.

We have provided an action recognition method based on the differential geometric properties of the volume generated from the object contours. The object volume can also be used for various other applications, such as object compression in MPEG-4 standard or synthesizing an action for different people. Another direction would be to explore volume features other than the features used in this thesis, such as first, second or third order moments for increasing the accuracy of the recognition performance.

LIST OF REFERENCES

- [AB87] J. Aloimonos and A. Badyopadhyay. “Active vision.” In *IEEE Int. Conf. on Computer Vision*, pp. 35–54, 1987.
- [AC99] J. K. Aggarwal and Q. Cai. “Human Motion Analysis: A Review.” *Computer Vision and Image Understanding*, **73**(3):428–440, 1999.
- [Aff] *Affine invariant interest point detector source code.*
<http://www.nada.kth.se/laptev/code.html>.
- [AKZ00] J.M. Appell, A.S. Kalitvin, and P.P. Zabrejko. *Partial integral operators and integro-differential equations*. M. Dekker, 2000.
- [Avi01] S. Avidan. “Support Vector Tracking.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.
- [BAH92] J. R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. “Hierarchical Model-based Motion Estimation.” In *European Conf. on Computer Vision*, pp. 237–252, 1992.
- [BB94] R.N. Braithwaite and B. Bhanu. “Hierarchical Gabor Filters for Object Detection in Infrared Images.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 628–631, 1994.
- [BBA00] S. Besson, M. Barlaud, and G. Aubert. “Detection and Tracking of Moving Objects Using a Level Set Based Method.” In *Int. Conf. of Pattern Recognition*, volume 3, pp. 1100–1105, 2000.
- [BC86] T.J. Broida and R. Chellappa. “Estimation of Object Motion Parameters from a Sequence of Noisy Images.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8**(1), 1986.
- [BD01] A. Bobick and J. Davis. “The Representation and Recognition of Action Using Temporal Templates.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**(3):257–267, 2001.
- [BF88] Y. Bar-Shalom and T.E. Foreman. *Tracking and Data Association*. Academic Press Inc., 1988.

- [BG89] J. Beaulieu and M Goldberg. “Hierarchy in picture image segmentation a step wise optimization approach.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11**:150–163, 1989.
- [Bir98] S. Birchfield. “Elliptical Head Tracking Using Intensity Gradients and Color Histograms.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 232–237, 1998.
- [BJ86] P. Besl and R. Jain. “Invariant surface characteristics for three dimensional object recognition in range images.” *Journal of Graphical Models and Image Processing*, **33**, 1986.
- [BJ98] M. Black and A. Jepson. “EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation.” *Int. Journal of Computer Vision*, **26**(1):63–84, 1998.
- [BK99] D. Beymer and K. Konolige. “Real-time tracking of multiple people using continuous detection.” In *ICCV: Frame-Rate Workshop*, 1999.
- [BL97] A. L. Blum and P. Langley. “Selection of relevant features and examples in machine learning.” *Artificial Intelligence*, **97**(1-2):245–271, 1997.
- [Bla92] M. Black. “Combining intensity and motion for incremental segmentation and tracking over long image sequences.” In *European Conf. on Computer Vision*, 1992.
- [BM92] P.J. Besl and N.D. McKay. “A registration of 3D shapes.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **14**(2):239–256, 1992.
- [Bov91] A.C. Bovik. “Analysis of multichannel narrow-band filters for image texture segmentation.” *IEEE Trans. on Image Processing*, **39**:2025–2043, 1991.
- [BSR00] M. Bertalmio, G. Sapiro, and G. Randall. “Morphing active contours.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(7):733–737, 2000.
- [BY92] A. Blake and A. Yuille. *Active vision*. Artificial Intelligence. MIT Press, 1992.
- [BY97] M.J. Black and Y. Yacoob. “Recognizing Facial Expressions in Image Sequences Using Local Parameterized Models of Image Motion.” *Int. Journal of Computer Vision*, **25**(1):23–48, 1997.
- [BZ92] A. Blake and A. Zisserman. “Surface Descriptions from Stereo and Shading.” *Image and Vision Computing Journal*, **3**(2):183–191, 1992.
- [CA91] Y. L. Chang and J. K. Aggarwal. “3d structure reconstruction from an ego motion sequence using statistical estimation and detection theory.” In *Workshop on Visual Motion*, 1991.

- [CA99] Q. Cai and J.K. Aggarwal. “Tracking human motion in structured environments using a distributed camera system.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **2**(11):1241–1247, 1999.
- [CH96] I.J. Cox and S.L. Hingorani. “An efficient implementation of Reid’s multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **18**(2):138–150, 1996.
- [Che95] Yizong Cheng. “Mean Shift, Mode Seeking, and Clustering.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **17**:790–799, 1995.
- [Chu02] T.J. Chung. *Computational fluid dynamics*. Cambridge University Press, 2002.
- [CKS95] V. Caselles, R. Kimmel, and G. Sapiro. “Geodesic active contours.” In *IEEE Int. Conf. on Computer Vision*, pp. 694–699, 1995.
- [CKS02] D. Cremers, T. Kohlberger, and C. Schnorr. “Non-linear shape statistics in Mumford-Shah based segmentation.” In *European Conf. on Computer Vision*, 2002.
- [CLF01] R.T. Collins, A.J. Lipton, H. Fujiyoshi, and T. Kanade. “Algorithms for cooperative multisensor surveillance.” *Proceedings of IEEE*, **89**(10):1456–1477, 2001.
- [CLL01] H. Chen, H. Lin, and T. Liu. “Multi-object tracking using dynamical graph matching.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 210–217, 2001.
- [CM99] D. Comaniciu and P. Meer. “Mean shift analysis and applications.” In *IEEE Int. Conf. on Computer Vision*, volume 2, pp. 1197–1203, 1999.
- [CM02] D. Comaniciu and P. Meer. “Mean Shift: A Robust Approach toward Feature Space Analysis.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(5):603–619, 2002.
- [Com02] D. Comaniciu. “Bayesian Kernel Tracking.” In *Annual Conf. of the German Society for Pattern Recognition*, pp. 438–445, 2002.
- [Cox93] I. J. Cox. “A Review of Statistical Data Association Techniques for Motion Correspondence.” *Int. Journal of Computer Vision*, **10**, 1993.
- [CR99] T.J. Cham and J. M. Rehg. “A Multiple Hypothesis Approach to Figure Tracking.” In *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 239–245, 1999.
- [CR00] H. Chui and A. Rangarajan. “A new algorithm for non-rigid point matching.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

- [CRH01] Y. Chen, Y. Rui, and T.S. Huang. “JPDAF Based HMM for Real-Time Contour Tracking.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.
- [CRM00] D. Comaniciu, V. Ramesh, and P. Meer. “Real-time Tracking of Non-rigid Objects Using Mean Shift.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pp. 142–149, 2000.
- [CRM03] D. Comaniciu, v. Ramesh, and P. Meer. “Kernel-Based object tracking.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **25**:564–575, 2003.
- [CS97] V. Caselles and G. Sapiro. “Minimal surfaces based object segmentation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19**(4):394–398, 1997.
- [CS03] D. Cremers and C. Schnorr. “Statistical shape knowledge in variational motion segmentation.” *Image and Vision Computing Journal*, **21**:77–86, 2003.
- [Cui99] O. Cuisenaire. *Distance transformations: fast algorithms and applications to medical image processing*. PhD Thesis, Universit catholique de Louvain, 1999.
- [DD03] D. Doermann and D. DeMenthon. “Video Retrieval using Spatio-Temporal Descriptors.” In *ACM Multimedia*, 2003.
- [DH73] R. Duda and P. Hart. *Pattern classification and scene analysis*. Wiley, 1973.
- [DT01a] S. Dockstader and A. M. Tekalp. “Multiple camera tracking of interacting and occluded human motion.” *Proceedings of the IEEE*, **89**:1441–1455, 2001.
- [DT01b] S. Dockstader and M. Tekalp. “On the tracking of articulated and occluded video object motion.” *Real time imaging*, **7**(5):415–432, 2001.
- [DV02] M.N. Do and M. Vetterli. “Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance.” *Int. Journal of Computer Vision*, **11**(2):146–158, 2002.
- [ED01] A.M. Elgammal and L.S. Davis. “Probabilistic framework for segmenting people under occlusion.” In *IEEE Int. Conf. on Computer Vision*, pp. 145–152, 2001.
- [EDH02] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. “Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance.” *Proc. of IEEE*, **90**(7), 2002.
- [FA91] W.T. Freeman and E.H. Adelson. “The Design and Use of Steerable Filters.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **13**(9):891–906, 1991.
- [FH75] K. Fukunaga and L.D. Hostetler. “The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition.” *IEEE IT*, **21**(1):32–40, 1975.

- [FT97] P. Fieguth and D. Terzopoulos. “Color-based tracking of heads and other mobile objects at video frame rates.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 21–27, 1997.
- [Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
- [Gav99] D. M. Gavrila. “The Visual Analysis of Human Movement: A Survey.” *Computer Vision and Image Understanding*, **73**(1):82–98, 1999.
- [GBC00] X. Gao, T.E. Boult, F. Coetzee, and V. Ramesh. “Error Analysis of Background Subtraction.” In *IEEE Int. Conf. on Computer Vision*, 2000.
- [GBG94] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C.H. Anderson. “Overcomplete steerable pyramid filters and rotation invariance.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 222–228, 1994.
- [GSR98] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. “Using adaptive tracking to classify and monitor activities in a site.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 22–29, 1998.
- [har] *Harris source code*. <http://www.cs.uwa.edu.au/~pk/Research/MatlabFns/Spatial/harris.m>.
- [HCP02] C. Hue, J. Le Cadre, and P. Prez. “Sequential Monte Carlo Methods for Multiple TargetTracking and Data Fusion.” *IEEE Trans. on Signal Processing*, **50**(2):309–325, 2002.
- [HE02] E. Hayman and J. Eklundh. “Probabilistic and voting approaches to cue integration for figure ground segmentation.” In *European Conf. on Computer Vision*, pp. 469–486, 2002.
- [HHD00] I. Haritaoglu, D. Harwood, and L.S. Davis. “W4: real-time surveillance of people and their activities.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(8):809–830, 2000.
- [Hir90] C. Hirsch. *Numerical Computation of Internal and External Flows, Volumes 1 and 2*. Wiley, 1990.
- [HO85] D. Howe and Others. *The Free On-line Dictionary of Computing*. <http://www.foldoc.org>, 1985.
- [HPB98] T. Hofmann, J. Puzicha, and J.M. Buhmann. “Unsupervised texture segmentation in a deterministic annealing framework.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **20**(8):803–818, 1998.

- [HR97] T. Huang and S. Russell. “Object Identification in a Bayesian Context.” In *Proceedings of IJCAI*, 1997.
- [HRC99] T. Huang, Y. Rui, and S.F. Chang. “Image Retrieval: Current Techniques, Promising Directions And Open Issues.” *Jrn. of Visual Comm. and Image Rep.*, **10**(4), 1999.
- [HS81] B.K.P. Horn and B.G. Schunck. “Determining optical flow.” *Artificial Intelligence*, **17**:185–203, 1981.
- [HS88] C.G. Harris and M. Stephens. “A combined corner and edge detector.” In *4th Alvey Vision Conference*, pp. 147–151, 1988.
- [HSD73] R. Haralick, B. Shanmugam, and I. Dinstein. “Textural features for image classification.” *IEEE Trans. on Systems, Man and Cybernetics*, **33**(3):610–622, 1973.
- [HZ99] A. Hanjalic and H. Zhang. “An Integrated Scheme for Automated Video Abstraction Based on Unsupervised Cluster-Validity Analysis.” *IEEE Trans. on Circuits and Systems*, **9**(8), 1999.
- [IA98a] M. Irani and P. Anandan. “Video Indexing Based on Mosaic Representations.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **20**(6):577–589, 1998.
- [IA98b] M. Irani and P. Anandan. “Video Indexing Based on Mosaic Representations.” *Proc. of IEEE*, **86**(5):905–921, 1998.
- [IB98] Michael Isard and Andrew Blake. “CONDENSATION - conditional density propagation for visual tracking.” *Int. Journal of Computer Vision*, **29**(1):5–28, 1998.
- [IDB97] S. Intille, J. Davis, and A. Bobick. “Real-time closed-world tracking.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 697–703, 1997.
- [IM01] M. Isard and J. MacCormick. “BraMBLe: A Bayesian Multiple-Blob Tracker.” In *IEEE Int. Conf. on Computer Vision*, 2001.
- [JF91] A.K. Jain and F. Farrokhnia. “Unsupervised texture segmentation using Gabor filters.” *Pattern Recognition Journal*, **24**(12):1167–1186, 1991.
- [JFE01] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. “Robust online appearance models for visual tracking.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pp. 415–422, 2001.
- [JFE03] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. “Robust Online Appearance Models for Visual Tracking.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **25**(10):1296–1311, 2003.

- [JN79] R. Jain and H.H. Nagel. “On the analysis of accumulative difference pictures from image sequences of real world scenes.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **1**(2):206–214, 1979.
- [JR02] M.J. Jones and J.M. Rehg. “Statistical Color Models with Application to Skin Detection.” *Int. Journal of Computer Vision*, **46**(1):81–96, 2002.
- [JRS03] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. “Tracking across multiple cameras with disjoint views.” In *IEEE Int. Conf. on Computer Vision*, 2003.
- [JS02] O. Javed and M. Shah. “Tracking and object classification for automated surveillance.” In *European Conf. on Computer Vision*, volume 4, pp. 343–357, 2002.
- [kal] *Kalman filtering source code.* <http://www.ai.mit.edu/~murphyk/Software/index.html>.
- [KCL98] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. “Advances in Cooperative Multi-Sensor Video Surveillance.” In *Darpa IU Workshop*, pp. 3–24, 1998.
- [KCM03] J. Kang, I. Cohen, and G. Medioni. “Continuous Tracking Within and Across Camera Streams.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.
- [klt] *KLT source code.* <http://robotics.stanford.edu/~birch/klt/>.
- [KP02] A. Kapoor and R.W. Picard. “Real-time, fully automatic upper facial feature tracking.” In *European Conf. on Computer Vision*, pp. 10–15, 2002.
- [KS01] S. Khan and M. Shah. “Object based segmentation of video using color motion and spatial information.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pp. 746–751, 2001.
- [KS03] S. Khan and M. Shah. “Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **25**, 2003.
- [Kuh55] H.W. Kuhn. “The Hungarian method for solving the assignment problem.” *Naval Research Logistics Quarterly*, **2**:83–97, 1955.
- [KWT88] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: active contour models.” *Int. Journal of Computer Vision*, **1**:321–332, 1988.
- [KZ99] V. Kettner and R. Zabih. “Bayesian multi-camera surveillance.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 117–123, 1999.

- [Lam96] S.W. Lam. “Texture feature extraction using gray level gradient based co-occurrence matrices.” In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 1, pp. 267–271, 1996.
- [Law80] K.I. Laws. *Textured image segmentation*. PhD Thesis, Electrical Eng., University of Southern California, 1980.
- [Leu01] T. Leung. “Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons.” *Int. Journal of Computer Vision*, **43**(1):29–44, 2001.
- [lev] *Level set source code*. <http://www.cs.utah.edu/whitaker/vispack/>.
- [LK81] Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision.” In *International Joint Conference on Artificial Intelligence*, 1981.
- [LL03] I. Laptev and T. Lindeberg. “Space-time interest points.” In *IEEE Int. Conf. on Computer Vision*, 2003.
- [LRS00] L. Lee, R. Romano, and G. Stein. “Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame.” *IEEE Trans. on Pattern Recognition and Machine Intelligence*, **22**(8):758–768, Aug 2000.
- [Mac98] D. J. C. MacKay. “Introduction to Monte Carlo Methods.” In M. I. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pp. 175–204. Kluwer Academic Press, 1998.
- [Mal89] S.G. Mallat. “A theory for multiresolution signal decomposition: the wavelet representation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11**(7):674–693, 1989.
- [Man02] A.R. Mansouri. “Region tracking via level set PDEs without motion computation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(7):947–961, 2002.
- [Mar82] D. Marr. *Vision*. W.H. Freeman and Co., 1982.
- [MB00] J. MacCormick and A. Blake. “Probabilistic exclusion and partitioned sampling for multiple object tracking.” *Int. Journal of Computer Vision*, **39**(1), 2000.
- [MD03] A. Mittal and L.S. Davis. “M2 tracker: a multi-view approach to segmenting and tracking people in a cluttered scene.” *Int. Journal of Computer Vision*, **51**(3):189–203, 2003.
- [meaa] *Mean-Shift segmentation source code*. <http://www.caip.rutgers.edu/riul/research/code.html>.

- [meab] *Mean-Shift tracking source code*. <http://www.intel.com/research/mrl/research/opencv/overview.htm>.
- [MG01] T.B. Moeslund and E. Granum. “A Survey of Computer Vision-Based Human Motion Capture.” *Computer Vision and Image Understanding*, 2001.
- [MMP03] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. “Background Modeling and Subtraction of Dynamic Scenes.” In *IEEE Int. Conf. on Computer Vision*, 2003.
- [Mor79] H.P. Moravec. “Visual mapping by a robot rover.” In *Proc. of IJCAI*, pp. 598–600, 1979.
- [MP97] Baback Mughadam and Alex Pentland. “Probabilistic Visual Learning for Object Representation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19**(7), 1997.
- [MS89] D. Mumford and J. Shah. “Optimal approximations by piecewise smooth functions and variational problems.” *Communication on Pure and applied Mathematics*, **42**(5):677–685, 1989.
- [MS02] K. Mikolajczyk and C. Schmid. “An affine invariant interest point detector.” In *European Conf. on Computer Vision*, volume 1, pp. 128–142, 2002.
- [MS03] K. Mikolajczk and C. Schmid. “A performance evaluation of local descriptors.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.
- [MSK89] L. Matthies, R. Szeliski, and T. Kanade. “Kalman Filter-based Algorithms for Estimating Depth from Image Sequences.” *Int. Journal of Computer Vision*, **3**(3), 1989.
- [MTH03] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. “Human body model acquisition and tracking using voxel data.” *Int. Journal of Computer Vision*, **53**(3):199–223, 2003.
- [Mur68] K.G. Murty. “An algorithm for ranking all the assignments in order of increasing cost.” *Operations research*, 1968.
- [NA94] S. A. Niyogi and E. H. Adelson. “Analyzing gait with spatiotemporal surfaces.” In *IEEE Workshop on Nonrigid and Articulated Motion*, pp. 64–69, 1994.
- [Ols00] C.F. Olson. “Maximum-likelihood template matching.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pp. 52–57, 2000.
- [ORP00] N.M. Oliver, B. Rosario, and A. Pentland. “A bayesian computer vision system for modeling human interactions.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(8):831–843, 2000.

- [par] *Particle filtering source code.* <http://www-sigproc.eng.cam.ac.uk/smc/smcsoftware.html>.
- [Pas01] G. Paschos. “Perceptually uniform color spaces for color texture analysis: an empirical evaluation.” *IEEE Trans. on Image Processing*, **10**:932–937, 2001.
- [PD00] N. Paragios and R. Deriche. “Geodesic active contours and level sets for the detection and tracking of moving objects.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(3):266–280, 2000.
- [PD02] N. Paragios and R. Deriche. “Geodesic Active Regions and Level Set Methods for Supervised Texture Segmentation.” *Int. Journal of Computer Vision*, **46**(3):223–247, 2002.
- [PS00] J. Portilla and E.P. Simoncelli. “A Parametric Texture Model based on Joint Statistics of Complex Wavelet Coefficients.” *Int. Journal of Computer Vision*, **40**(1):49–70, 2000.
- [RB96] S. Rowe and A. Blake. “Statistical Mosaics For Tracking.” *Image and Vision Computing Journal*, **14**:549–564, 1996.
- [Rei79] D. B. Reid. “An Algorithm for Tracking Multiple Targets.” *IEEE Transactions on Automatic Control*, **24**(6):843–854, 1979.
- [RH01] C. Rasmussen and G. Hager. “Probabilistic Data association methods for tracking complex visual objects.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**(6):560–576, 2001.
- [RKJ00] J. Rittscher, J. Kato, S. Joga, and A. Blake. “A Probabilistic Background Model for Tracking.” In *European Conf. on Computer Vision*, volume 2, pp. 336–350, 2000.
- [Ron94] R. Ronfard. “Region based strategies for active contour models.” *Int. Journal of Computer Vision*, **13**(2):229–251, 1994.
- [RS91] K. Rangarajan and M. Shah. “Establishing motion correspondence.” *Computer Vision, Graphics and Image Processing*, **54**(1):56–73, 1991.
- [RS99] R. Rosales and S. Sclaroff. “3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.
- [RS03] Z. Rasheed and M. Shah. “Scene detection in hollywood movies and tv shows.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, p. in press, 2003.
- [RY03] R. Collins and Y. Liu. “On-Line Selection of Discriminative Tracking Features.” In *IEEE Int. Conf. on Computer Vision*, 2003.

- [SB97] S.M. Smith and J.M. Brady. “SUSAN - a new approach to low level image processing.” *Int. Journal of Computer Vision*, **23**(1):45–78, 1997.
- [SBL02] J. Smith, S. Basu, C. Ling, M.Naphade, and B. Tseng. “Interactive Content-Based Video Retrieval.” In *ICIP*, 2002.
- [Sco92] D.W. Scott. *Multivariate density estimation*. Wiley, 1992.
- [SD97] S.M. Seitz and C.R. Dyer. “View-invariant analysis of cyclic motion.” *Int. Journal of Computer Vision*, **25**:1–25, 1997.
- [Set99] J.A. Sethian. *Level set methods: evolving interfaces in geometry, fluid mechanics computer vision and material sciences*. Cambridge University Press, 1999.
- [SG00] C. Stauffer and W.E.L. Grimson. “Learning patterns of activity using real time tracking.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(8):747–767, 2000.
- [SJ87] I.K. Sethi and R. Jain. “Finding trajectories of feature points in a monocular image sequence.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **9**(1):56–73, 1987.
- [SKP96] K. Y. Song, J. Kittler, and M. Petrou. “Defect detection in random color textures.” *Image and Vision Computing Journal*, **14**(9), 1996.
- [SL94] R. L. Streit and T. E. Luginbuhl. “Maximum Likelihood method for probabilistic multi-hypothesis tracking.” In *proceedings of SPIE*, volume 2235, 1994.
- [SM00] J. Shi and J. Malik. “Normalized cuts and image segmentation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22**(8):888–905, 2000.
- [SP96] T. Starner and A. Pentland. “Real-time American sign language recognition from video using hidden Markov models.” In *Motion- Based Recognition*. Kluwer, 1996.
- [SRP01] Bjoern Stenger, Visvanathan Ramesh, Nikos Paragios, Frans Coetzee, and Joachim M. Buhmann. “Topology Free Hidden Markov Models: Application to Background Modeling.” In *IEEE Int. Conf. on Computer Vision*, 2001.
- [SS90] V. Salari and I. K. Sethi. “Feature point correspondence in the presence of occlusion.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **12**(1):87–91, 1990.
- [SS93] M. Swain and M. Stricker. “Promising directions in active vision.” *Int. Journal of Computer Vision*, **11**:109–126, 1993.
- [SS03] K. Shafique and M. Shah. “A Non-Iterative Greedy Algorithm for Multi-frame Point Correspondence.” In *IEEE Int. Conf. on Computer Vision*, 2003.

- [ST94] J. Shi and C. Tomasi. “Good features to track.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.
- [sus] *SUSAN source code*. <http://www.fmrib.ox.ac.uk/~steve/susan>.
- [TAE98] M. Tekalp, Y. Altunbasak, and E. Eren. “Region based parametric motion segmentation using color information.” *Journal of Graphical Models and Image Processing*, **60**(1):13–23, 1998.
- [Tan87] H. Tanizaki. “Non-Gaussian State-Space Modeling of Nonstationary Time Series.” *Journal of the American Statistical Association*, **82**:1032–1063, 1987.
- [TB02] L. Torresani and C. Bregler. “Space-Time Tracking.” In *European Conf. on Computer Vision*, 2002.
- [TJM99] K. Toyama, B. Brumitt, J. Krumm, and B. Meyers. “Wallflower: Principles and Practices of Background Maintenance.” In *IEEE Int. Conf. on Computer Vision*, 1999.
- [TS92] D. Terzopoulos and R. Szeliski. “Tracking with Kalman Snakes.” In A. Blake and A. Yuille, editors, *Active Vision*. MIT Press, 1992.
- [TSA01] P. Torr, R. Szeliski, and P. Anandan. “An Integrated Bayesian Approach to Layer Extraction from Image Sequences.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**(3):297–303, 2001.
- [TSK02] H. Tao, H.S. Sawhney, and R. Kumar. “Object tracking with bayesian estimation of dynamic layer representations.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(1):75–89, 2002.
- [TV04] K. Tieu and P. Viola. “Boosting Image Retrieval.” *Int. Journal of Computer Vision*, **56**(1):17–36, 2004.
- [Vap98] V. Vapnik. *Statistical Learning Theory*. Wiley NY., 1998.
- [Vit67] A. J. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.” *IEEE Trans. Inform. Theory*, **13**:260–269, 1967.
- [VRB01] C.J. Veenman, M.J. Reinders, and E. Backer. “Resolving Motion Correspondence for Densely Moving Points.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**(1):54–72, 2001.
- [VRC03] N. Vaswani, A. RoyChowdhury, and R. Chellappa. “Activity Recognition Using the Dynamics of the Configuration of Interacting Objects.” In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

- [WAP97] C.R. Wren, A. Azarbayejani, and A. Pentland. “Pfinder: Real-Time Tracking of the Human Body.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19**(7):780–785, 1997.
- [Wei02] E.W. Weisstein. *CRC Concise Encyclopedia of Mathematics, Second Edition*. CRC Press, 2002.
- [WL93] Z. Wu and R. Leahy. “An optimal graph theoretic approach to data clustering: Theory and its applications to image segmentation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11**:1101–1113, 1993.
- [WWJ03] G. Wu, Y. Wu, L. Jiao, Y.F. Wang, and E.Y. Chang. “Multi-camera Spatio-temporal Fusion and Biased Sequence-data Learning for Security Surveillance.” In *ACM Multimedia*, 2003.
- [XA02] N. Xu and N. Ahuja. “Object contour tracking using graph cuts based active contours.” In *IEEE Int. Conf. on Image Processing*, 2002.
- [YLS04] A. Yilmaz, X. Li, and M. Shah. “Object tracking using level sets.” In *Asian Conf. on Computer Vision*, 2004.
- [YRS02] A. Yilmaz, C. Rao, and M. Shah. “View invariant representation and recognition of actions.” *Int. Journal of Computer Vision*, **50**(2):203–226, 2002.
- [YS00] A. Yilmaz and M. Shah. “Shot Detection Using Principal Coordinate System.” In *IASTED Internet and Multimedia Systems and Applications Conf.*, p. 168, 2000.
- [YS02a] A. Yilmaz and M. Shah. “Automatic Feature Detection and Pose Recovery for Faces.” In *Asian Conf. on Computer Vision*, pp. 284–289, 2002.
- [YS02b] A. Yilmaz and M. Shah. “Estimation of Arbitrary Albedo and Shape from Shading for Symmetric Objects.” In *British Machine Vision Conference*, pp. 728–736, 2002.
- [YS04] A. Yilmaz and M. Shah. “Contour Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2004.
- [YSL01] A. Yilmaz, K.H. Shafique, N.V Lobo, X. Li, T. Olson, and M. Shah. “Target-Tracking in FLIR Imagery Using Mean-Shift and Global Motion Compensation.” In *IEEE CVPR Workshop on Computer Vision Beyond Visible Spectrum*, 2001.
- [YSS02] A. Yilmaz, K. Shafique, and M. Shah. “Estimation of rigid and nonrigid motion using anatomical face model.” In *Int. Conf. of Pattern Recognition*, volume 1, pp. 377–380, 2002.

- [YSS03] A. Yilmaz, K. Shafique, and M. Shah. “Target Tracking in Airborne Forward Looking Imagery.” *Journal of Image and Vision Computing*, **21**(7):623–635, 2003.
- [ZCM03] S. Zhou, R. Chellapa, and B. Moghadam. “Adaptive Visual Tracking and Recognition using Particle Filters.” In *Proc. of ICME*, 2003.
- [ZS03] J. Zhong and S. Sclaroff. “Segmenting Foreground Objects from a Dynamic Textured Background via a Robust Kalman Filter.” In *IEEE Int. Conf. on Computer Vision*, 2003.
- [ZY96] S.C. Zhu and A. Yuille. “Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **18**(9):884–900, 1996.