

VIDEO OBJECT TRACKING WITH FEEDBACK OF PERFORMANCE
MEASURES

by

Çiğdem Eroğlu Erdem

B.S. in Electrical and Electronics Engineering, Bilkent University, 1995

M.S in Electrical and Electronics Engineering, Bilkent University, 1997

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor
of
Philosophy

Boğaziçi University

2002

VIDEO OBJECT TRACKING WITH FEEDBACK OF PERFORMANCE
MEASURES

APPROVED BY:

Prof. Bülent Sankur
(Thesis Supervisor)

Prof. Emin Anarım
(Thesis Co-supervisor)

Assoc. Prof. Lale Akarun

Prof. Muhittin Gökmen

Prof. A. Murat Tekalp

DATE OF APPROVAL: 21.06.2002

ACKNOWLEDGEMENTS

I gratefully thank my supervisor Prof. Bülent Sankur for his suggestions, supervision and guidance throughout the development of this thesis. I would also like to thank Prof. A. Murat Tekalp and Prof. Emin Anarım for their suggestions and valuable discussions.

I would like to thank Prof. Lale Akarun and Prof. Muhittin Gökmen, the members of my jury, for reading and commenting on the thesis.

It is a pleasure to express my special thanks to my family for their encouragement, love and patience.

ABSTRACT

VIDEO OBJECT TRACKING WITH FEEDBACK OF PERFORMANCE MEASURES

The task of segmentation and tracking of objects in a video sequence is an important high-level video processing problem for object-based video manipulation and representation. This task involves utilization of many low-level pre-processing tasks such as image segmentation and motion estimation. It is also very important to assess the performance of the video object segmentation and tracking algorithms quantitatively and objectively. Performance evaluation measures are proposed both when the ground-truth segmentation maps are available and when they are unavailable. A semi-automatic video object tracking method is introduced that uses the proposed performance evaluation measures in a feedback loop to adjust its parameters locally on the object boundary. New low-level image segmentation and motion estimation algorithms, namely, an illumination invariant fuzzy image segmentation algorithm and a motion estimation algorithm in the frequency domain using fuzzy c-planes clustering are also presented in this thesis.

ÖZET

VIDEO NESNELERİNİN BAŞARIM GERİBESLEMELİ İZLENMESİ

Video dizilerindeki nesnelere bölütlenme ve izlenme, nesne tabanlı video uygulamaları için önemli bir problemdir. Bu kapsamlı problem, imge bölütleme, kenar ve hareket kestirimi gibi daha başka ara basamakların çözümünü gerektirebilir. Nesne bölütleme ve izleme sonuçlarının nicel ve nesnel olarak değerlendirilmesi de çok önemli bir sorundur. Bu tezde, nesne bölütleme algoritmalarının başarımlarını nicel olarak değerlendirmek için ölçütler önerilmiştir. Bu ölçütlerin bir kısmı, gerçek referans bölüt haritalarını kullanır, diğer bir kısmı ise gerçek bölüt haritalarına ihtiyaç duymaz. Önerilen nesne izleme algoritması, bu başarımları ölçütlerini geribesleme olarak kullanarak, algoritma parametrelerini nesne sınırı boyunca yerel olarak değiştirir. Bulanık c-düzlemler kümeleme yöntemini kullanan iki imge bölütleme ve hareket kestirimi yöntemi de geliştirilmiştir. Bunlardan ilki, imge renk bileşenlerinden aydınlık bileşenine bulanık c-düzlemler oturtarak, yerel aydınlık değişimlerine duyarsızlık sağlayan bir imge bölütleme algoritmasıdır. İkinci yöntem ise, 3-boyutlu sıklık düzlemi verilerinin, bulanık c-düzlemler metodu ile düzlemlere yerleştirilmesine dayalı bir hareket kestirme algoritmasıdır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	ix
LIST OF TABLES	xv
LIST OF SYMBOLS/ABBREVIATIONS	xvii
1. INTRODUCTION	1
1.1. Motivation	1
1.2. Contribution and Scope	3
1.3. Literature Review	4
1.3.1. Video Object Segmentation and Tracking Techniques	4
1.3.1.1. Fully Automatic Techniques	5
1.3.1.2. Semi-Automatic (User Interactive) Techniques	15
1.3.2. Still Image Segmentation Techniques	17
2. LOW-LEVEL IMAGE AND VIDEO PROCESSING TECHNIQUES USING FUZZY PROCESSING	20
2.1. Illumination Invariant Fuzzy Image Segmentation	20
2.1.1. Image Segmentation Using Fuzzy c-planes Clustering	21
2.1.2. Extension to Color Images	25
2.1.3. Experimental Results	28
2.2. Motion Estimation in the Frequency Domain Using Fuzzy C-Planes Clustering	33
2.2.1. Theoretical Background	34
2.2.2. Motion Parameter Estimation Using Fuzzy C-Planes Clustering	36
2.2.3. Experimental Results	40
3. PERFORMANCE EVALUATION MEASURES FOR OBJECT-BASED VIDEO SEGMENTATION	49
3.1. Performance Evaluation Measures Based on Ground-Truth Data	49
3.1.1. Misclassification Penalty (DP).	50

3.1.2.	Shape Penalty (DS)	52
3.1.3.	Motion Penalty (DM)	55
3.1.4.	Combined Penalty (CP)	55
3.1.5.	Performance Evaluation of the Video Segmentation Using Ground- Truth Segmentation Maps	56
3.2.	Performance Evaluation Measures Without Ground-Truth Data	61
3.2.1.	Color Measures	61
3.2.1.1.	Intra-frame Color Differences Along the Object Boundary	62
3.2.1.2.	Inter-frame Color Histogram Differencing	64
3.2.2.	Motion Measure	66
3.2.3.	Performance Evaluation of Video Segmentation with Non-Ground- Truth Measures	68
3.2.3.1.	Combining Color and Motion Measures	68
3.2.3.2.	Temporal Localization	69
3.2.3.3.	Spatial Localization	69
3.2.4.	Experimental Results with Measures	70
3.2.4.1.	Experiments with Color Differences Along the Boundary	70
3.2.4.2.	Experiments with Color Histogram Differences	71
3.2.4.3.	Experiments with the Motion Differences	71
3.2.4.4.	Localization of Incorrect Segmentation	72
3.3.	Comparing the Ground-Truth and Non-Ground-Truth Measures	76
3.4.	Statistical Analysis of Ground-Truth and Non-Ground-Truth Measures	78
3.4.1.	Principal Component Analysis of Performance Measures	79
3.4.2.	Canonical Correlation Analysis of Performance Measures	86
4.	VIDEO OBJECT TRACKING WITH FEEDBACK OF PERFORMANCE MEASURES	90
4.1.	Overview of The Framework	90
4.2.	Open-Loop Boundary Prediction	92
4.2.1.	Boundary Initialization	93
4.2.2.	Boundary Prediction	93
4.2.3.	Occlusion Handling	95
4.2.3.1.	Temporal Prediction	96

4.2.3.2.	Spatial Prediction	96
4.2.3.3.	Search Around The Predicted Location	97
4.3.	Closed-Loop Boundary Correction	97
4.3.1.	External Energy Terms	99
4.3.1.1.	Color Boundary Energy	100
4.3.1.2.	Edge Boundary Energy	101
4.3.1.3.	Motion Boundary Energy	102
4.3.2.	Internal Energy Term	102
4.3.3.	Selection of Search Directions	103
4.4.	Adaptation of Weights Using Performance Scores	104
4.4.1.	Reliability Assessment of Color and Motion Boundaries	105
4.4.2.	Weight Adaptation	106
4.5.	Experimental Results	108
4.5.1.	Results for the “Bream” Sequence	108
4.5.2.	Results for the “Parrot” Sequence	110
4.5.3.	Results for the “Coastguard” Sequence	111
5.	CONCLUSIONS AND FUTURE DIRECTIONS	119
5.1.	Conclusions	119
5.2.	Future Directions	121
	REFERENCES	122

LIST OF FIGURES

Figure 1.1.	The simplified flowchart of the Altunbaşak-Eren-Tekalp method with the improved steps shown with shaded boxes	11
Figure 1.2.	The simplified flowchart of the Meier-Ngan method with the improved steps shown with shaded boxes	12
Figure 1.3.	The simplified flowchart of the Castagno-Ebrahimi-Kunt method	13
Figure 2.1.	(a) The original image, PSNR = 10dB (b) The result using the FCM algorithm (c) The result using the proposed FCP algorithm, where gray levels for the class centers are: 147, 172, 248, 7	30
Figure 2.2.	(a) The original image, PSNR = 4dB (b) The result using the FCM algorithm (c) The result using the proposed FCP algorithm	30
Figure 2.3.	(a) The original image, PSNR = 10dB (b) The result using the FCM algorithm (c) The result using the proposed FCP algorithm	30
Figure 2.4.	(a) Original image, PSNR = 4dB (b) Segmentation result of the fuzzy c-means algorithm (c) Segmentation result of the proposed fuzzy c-planes algorithm	31
Figure 2.5.	(a) Original image PSNR = 0dB (b) Segmentation result of the FCM algorithm (c) Segmentation result of the proposed FCP algorithm	31
Figure 2.6.	(a) The original ROBOT image (b) Segmentation result of the FCM algorithm (c) Segmentation result of the proposed FCP algorithm	32

Figure 2.7.	(a) The original COLA CAN image (b) Segmentation result of the FCM algorithm (c) Segmentation result of the proposed FCP algorithm	32
Figure 2.8.	The 1 st and the 20 th frames of the sequence containing three moving blocks	42
Figure 2.9.	The theoretical motion planes for the sequence containing three moving blocks	43
Figure 2.10.	The periodogram plots for (a) $(\omega_x, \omega_y) = (0.127, 0.698)$, (b) $(\omega_x, \omega_y) = (0.508, 0.508)$, (c) $(\omega_x, \omega_y) = (0, 0.508)$, (d) $(\omega_x, \omega_y) = (0.508, 0)$	44
Figure 2.11.	The data points clustered into planes using the FCP algorithm	45
Figure 2.12.	The motion planes estimated using the clustered data shown in Figure 2.11	46
Figure 2.13.	The FCP cost function versus number of planes. The magnitudes have been scaled so that the maximum cost value is mapped to 1	47
Figure 2.14.	The 1 st and the 20 th frames of the “Taxi in garden” sequence	47
Figure 2.15.	The 1 st and the 20 th frames of the “Hamburg Taxi” sequence	48
Figure 2.16.	The 130 th and the 140 th frames of the “Coast Guard” sequence	48
Figure 3.1.	The turning angle function of a square	53
Figure 3.2.	The turning angle function of a fish boundary	54

Figure 3.3.	(a) The second frame of the “Taxi in the garden” sequence (b) The fourth frame of the “Hamburg taxi” sequence	58
Figure 3.4.	(a) The ground truth VOP for frame 2 of the “Taxi in the garden” sequence. (b), (c), (d) The segmentation results of Method 1, 2, and 3, respectively	59
Figure 3.5.	(a) The ground truth VOP for the fourth frame of the “Hamburg taxi” sequence (b), (c), (d) The segmentation results of Method 1, 2, and 3, respectively	59
Figure 3.6.	(a) The misclassification penalty $DP(t)$ (b) The shape penalty $DS(t)$ (c) The motion penalty $DM(t)$ for the “Taxi in the garden sequence”	60
Figure 3.7.	(a) A sample VOP (b) The boundary of the VOP with the normal lines (c) A close-up of a normal line drawn to the boundary	63
Figure 3.8.	A sample S function, $p_1 = 0.2, p_2 = 0.5, p_3 = 0.8$	69
Figure 3.9.	Frames 32 and 230 of the “Hall Monitor” sequence	70
Figure 3.10.	The color histogram differences between H_t and $H_{t,av}$, calculated with χ^2 measure, where segmentation maps are shifted by ± 10 pixels, starting from frame 100	72
Figure 3.11.	(a) The video object plane for the 134 th frame (b) Incorrectly segmented regions are marked with black boxes	74
Figure 3.12.	The color differences $d_{CB}(t; i)$ along the boundary of segmented Bream object	74

Figure 3.13.	(a) Segmentation of the 123 th frame (b) Incorrectly segmented regions using the color difference measure (b) Incorrectly segmented regions using the motion difference measure (b) Final decision of incorrectly segmented regions	75
Figure 3.14.	Measures with ground-truth: (a) The misclassification penalty $DP(t)$ for each frame (b) The shape penalty $DS(t)$ (c) The motion penalty $DM(t)$	83
Figure 3.15.	Measures without ground-truth: (a) Color differences along boundary (b) Motion differences along boundary (c), (d), (e), and (f) show the inter-frame histogram differences using L_1 , L_2 , χ^2 and the histogram intersection, respectively	84
Figure 3.16.	(a) Scatter plot of $DP(t)$ versus $DS(t)$ (b) $DP(t)$ versus $d_{CB}(t)$ (c) $DP(t)$ versus $d_{CHHI}(t)$ (d) $d_{CHL_1}(t)$ versus $d_{CHHI}(t)$ (e) The scatter plot of the first pair of canonical variables	85
Figure 4.1.	The tracking framework consists of the <i>boundary prediction</i> and the <i>boundary correction</i> steps	92
Figure 4.2.	(a) Locally adaptive boundary prediction using feature point tracking (b) A magnified view of the section of the boundary shown in the box in (a)	93
Figure 4.3.	(a) The first frame of “Bream” (b) Two class color segmentation (c) Color segmentation boundaries (d) The Chamfer 3-4 transform of color segmentation boundaries (e) The edges	99
Figure 4.4.	Illustration of search locations at each node during dynamic programming	101

- Figure 4.5. Illustration of energy minimization using dynamic programming. The correct and estimated (to be updated) object boundaries are shown by the thin and thick lines, respectively 101
- Figure 4.6. (a) The predicted boundary to be corrected is shown by red points. The selective search directions are shown by green lines (b) A close-up of (a) 104
- Figure 4.7. (a) Frame 127 of the “Bream” sequence (b) The color segmentation boundaries (c) The color segmentation reliability map in which dark regions indicate unreliable color segmentation boundaries 105
- Figure 4.8. Row 1: Hand-drawn boundary at frame 100. Row 2 - Row 5: Tracking results using open-loop, and closed loop with edge energy only, equal weighting and adaptive weighting methods, respectively, for frames 109, 121, and 128 113
- Figure 4.9. (a) The number of misclassified pixels for each frame of ‘Bream’ (b) The color measure along the estimated boundary for each frame (c) The average distance from actual object boundary for each frame 114
- Figure 4.10. Row 1: Hand-drawn boundary at frame 1. Row 2 - Row 4: Tracking results using closed-loop with edge energy only, equal weighting and adaptive weighting methods, respectively, for frames 2, 8, and 18 115
- Figure 4.11. (a) Edges of the second frame (b) Color segmentation boundaries (c) Color-refined motion segmentation boundaries (d) Green segments: zero motion energy weight. White segments: weight of the motion energy term is the largest 116

- Figure 4.12. (a) The number of misclassified pixels for “Parrot” sequence (b) Color difference measure along boundary (c) Average distance from the actual object boundary for each frame 117
- Figure 4.13. Frames 1, 45, 72, and 86 of the “Coast Guard” sequence 118
- Figure 4.14. Open-loop tracking results for frames 1, 45, 72, and 86. Tracked and predicted feature points are shown by yellow and red boxes, respectively. Motion vectors are shown by blue lines 118

LIST OF TABLES

Table 1.1.	Some of the video object segmentation/tracking methods in the literature	14
Table 2.1.	The original and estimated motion vectors for the sequence containing three moving blocks	42
Table 2.2.	The original and estimated motion vectors for the “Hamburg Taxi” sequence	42
Table 2.3.	The original and estimated motion vectors for the “Coast Guard” sequence	42
Table 3.1.	The performances of the three object-based video segmentation methods averaged over all frames	58
Table 3.2.	The scores for color difference measure along the object boundary .	72
Table 3.3.	The scores for color histogram difference measure	73
Table 3.4.	The scores for motion difference measure along the object boundary	73
Table 3.5.	The mean of performance evaluation scores for each frame of the “Bream” sequence	77
Table 3.6.	The standard deviation of the performance evaluation scores for each frame of the “Bream” sequence. The mean values are given in Table 3.5	77
Table 4.1.	Summary of quantitative comparisons for the “Bream” sequence .	112

Table 4.2.	Summary of quantitative comparisons for the “Parrot” sequence	. 112
------------	---	-------

LIST OF SYMBOLS/ABBREVIATIONS

\mathbf{A}_i^t	The transformation matrix of boundary segment \mathbf{s}_i^t
$(\mathbf{a}_i, \mathbf{b}_i)$	The i^{th} pair of canonical linear transformation parameters
a_i, b_i, d_i	Parameters of the i^{th} plane
B	Total number of bins in the histogram
\mathbf{B}^t	The $2 \times N_t$ matrix of boundary pixels at frame t .
C	Number of classes in fuzzy clustering
\mathbf{c}_i	Centroid of i^{th} cluster
$\text{Cham}_{g_i}(x, y)$	Modified Chamfer distance of boundary of g_i
$\text{Col}_I^i(t)$	The average color around point $\mathbf{p}_I^i(t)$
$\text{Col}_O^i(t)$	The average color around point $\mathbf{p}_O^i(t)$
D_{CB}	The average of $d_{CB}(t)$ values for the video sequence
$d_{CB}(t)$	Measure of color differences along the estimated boundary
$d_{CB}(t; i)$	The color difference at normal line i
D_{CH}	The average histogram difference measure for the sequence
d_{HL_1}	The histogram difference calculated using the L_1 metric
d_{HL_2}	The histogram difference calculated using the L_2 metric
$d_{H_{\chi^2}}$	The histogram difference calculated using the χ^2 metric
d_{HHI}	The histogram difference calculated using the HHI metric
D_i	Distance to the i^{th} cluster.
D_{ij}	Distance between the i^{th} data vector and the j^{th} plane
D_M	Average motion measure for the whole sequence
\overline{DM}	Overall motion penalty of the sequence
$DM_i(t)$	The motion penalty of object i at frame t
$d_M(t; i)$	The motion difference at the i^{th} normal at frame t
$d_M(t)$	The measure of average motion differences at frame t
\overline{DP}	Overall misclassification penalty of the sequence
$DP_i(t)$	Misclassification penalty of object i at time t
$DP(t)$	Total misclassification penalty for frame t
\overline{DS}	Overall shape penalty of the sequence

$DS_i(t)$	Shape penalty of object i at frame t
$DS(t)$	Total shape penalty for frame t
$d(\mathbf{v}_o^i(t), \mathbf{v}_I^i(t))$	Distance between the two average motion vectors
\mathbf{e}_{ik}	k^{th} eigenvector of covariance matrix \mathbf{R}_i
\mathbf{e}_i	The i^{th} eigenvector of the covariance matrix $\mathbf{\Sigma}$
E_{col}	The energy from color segmentation boundaries
E_{curv}	The curvature energy.
$E_{ext,i}$	External energy of the snake for the i^{th} segment
E_{edge}	The edge energy
$E_{int,i}$	Internal energy of the snake for the i^{th} segment
E_{mot}	The energy from motion segmentation boundaries
E_{snake}	Energy of the snake.
\bar{f}_i	Average gray level of the i^{th} cluster
\mathbf{F}^t	The $2 \times M_t$ matrix of feature points close to the boundary.
$f_{Ii}(x, y)$	Ideal image of object i at time t
$f(x, y; t)$	Image sequence
$f(x, y)$	Image intensity value at pixel location (x, y)
$F(\omega_x, \omega_y; t)$	Fourier transform of $f(x, y; t)$
G	Object set of the ground-truth image
g_i	A ground truth object
\bar{h}_i	Cluster centroid of the hue values
\mathbf{h}_i	3D data obtained from periodogram peaks
H_t	Color histogram of the VOP at time t
$H_{t,av}$	Average color histogram of VOP's around time t
$h(x, y)$	Hue value of pixel at location (x, y)
$I_i(x, y; t)$	Indicator function of support region of object i .
J_{FCP}	Total energy in fuzzy c-planes clustering
K_t	The number of normal lines drawn to the estimated boundary
$L_G(x, y; t)$	Label function denoting the object index that the pixel at (x, y) belong to at time t in the ground truth frame.
$L_S(x, y; t)$	Label function denoting the object index that the pixel at (x, y) belong to at time t in the segmented frame.

M	Number of independently moving objects in the image
m_{ij}	Membership value of the i^{th} data vector to the j^{th} plane
$m_i(x, y)$	The membership value of pixel at (x, y) to class i
$\hat{m}_i(x, y)$	Estimated membership value of the pixel to class i
N	Number of pixels in a frame
$N_b(t)$	Number of boundary pixels at frame t
$N_f(t)$	Number of feature points close to the boundary at frame t
N_{H_t}	Total population of H_t
\mathbf{n}_i	The selected nodes on the boundary
N_ζ	The number of pixels in the neighborhood
NS_{H_t}	Sum of squared populations in each bin of the histogram
PC_i	The principal components of the data
$P_F(\omega_x, \omega_y; \omega)$	Periodogram of $F(\omega_x, \omega_y; t)$
$\mathbf{p}_b^i(t)$	Location of the i^{th} boundary pixel at frame t
$\mathbf{p}_f^i(t)$	Location of the i^{th} feature point at frame t
$\mathbf{p}_I^i(t)$	The pixel at the end of the i^{th} normal line inside the object boundary
$\mathbf{p}_O^i(t)$	The pixel at the end of the i^{th} normal line outside the object boundary
q	Fuzzification parameter
r_1, r_2	Ratio of total histogram populations
\mathbf{R}_j	Covariance matrix of the j^{th} plane
$Rel(\mathbf{v}_o^i(t))$	Reliability of the motion vector $\mathbf{v}_o^i(t)$
S	Object set of the segmented image
s_i	A segmented object
\bar{s}_i	Cluster centroid of the saturation values
\mathbf{s}_i^t	The submatrix of pixel locations of boundary segment i
$s(x, y)$	<i>Saturation</i> value of pixel at location (x, y)
T	Number of frames in a sequence
\tilde{u}_i	Member of the set of candidate motion parameters in x direction

\tilde{v}_i	Member of the set of candidate motion parameters in y direction
$(u \widetilde{+} v)_i$	Member of the set of sum of candidate motion parameters
(u_i, v_i)	Translational motion vector of object i
(U_i, V_i)	The i^{th} canonical variate pair
$v(x, y)$	<i>Value</i> value of pixel at location (x, y)
$\mathbf{v}_I^i(t)$	The average motion vector around $\mathbf{p}_I^i(t)$
$\mathbf{v}_O^i(t)$	The average motion vector around $\mathbf{p}_O^i(t)$
\bar{w}_{i0}	Average support window of object i over T frames
$W_i(t)$	Support window of object i
$W(t)$	Region of interest of the image at time t
X	Data matrix of ground truth performance scores
\mathbf{x}_i	Data vector of ground truth performance scores
(x, y)	Image pixel coordinates
(\bar{x}_i, \bar{y}_i)	Coordinates of the i^{th} cluster centroid
\mathbf{y}_i	Data vector of non-ground truth performance scores
Y	Data matrix of non-ground truth performance scores
\mathbf{z}_i	Data vector of all performance scores
Z	Data matrix of all performance scores
β_{col}	The weight of the color boundary energy term
β_{curv}	The weight of the curvature energy term
β_{edge}	The weight of the edge energy term
β_{mot}	The weight of the motion energy term
$\varepsilon(x, y; t)$	Error
η_i	The reliability weight (multiplication of two reliabilities)
λ_i	The i^{th} eigenvalue of the covariance matrix Σ
ω_l	l^{th} harmonic
Σ	The covariance matrix of standardized performance scores
$\xi(x, y; t)$	Difference between current and average support windows
FCP	Fuzzy C-Planes

GT	Ground Truth
PCA	Principal component analysis
VOP	Video object plane
VO	Video Object
WGT	Without Ground Truth

1. INTRODUCTION

1.1. Motivation

In the past few decades, many developments have occurred in the image and video processing technology, notably with the standardization efforts of the MPEG (Moving Picture Experts Group). Towards the end of 80's, MPEG-1 video coding standard enabled us to store moving pictures and audio on the compact disc. Then, with the help of MPEG-2, the traditional analog television broadcasting system was converted to digital. In the digital television technology at that stage, user interaction was enabled to a limited extent [1].

Today, while the MPEG-2 standard enhances the traditional ways of consuming video and audio, the available Internet seems to promise new ways of user interaction with video and audio content. The most important limitation to the development of interactive audio-visual services is the bandwidth restriction. If we want to open the capabilities of the web to interactive video, we have to go beyond the interpretation of video as a sequence of rectangular arrays of pixels as done in MPEG-1 and MPEG-2. If the coding algorithm has access to the semantic meaning of the “objects” composing the scene in a video sequence, using a defined “scene description”, it will be easier to manipulate and recompose a scene. A new MPEG standard called MPEG-4 provides standard ways to encode video and audio objects, scene descriptions, and the interface to the delivery system [1]. Representing the information with this new approach not only provides better interactivity, but also enables re-use of data, intelligent management of bandwidth, and error protection [2]. The interactivity enables us to [3]: place media objects anywhere in a given coordinate system; apply transforms to change the geometrical or acoustical appearance of a media object; group media objects in order to form compound media objects; apply streamed data to media objects, in order to modify their attributes (e.g. moving texture belonging to an object, animation parameters animating a moving head); interactively change the user's viewing and listening points anywhere in the scene; drag objects in the scene to a different position; trigger

a cascade of events by clicking on a specified object, e.g. starting or stopping a video stream.

The MPEG-4 standard does not describe how a given scene is decomposed or segmented into “objects”. This is because MPEG-4 is a decoding standard, like MPEG-1 and MPEG-2, and does not specify how the encoding will be done. In fact, the segmentation of a scene into its component “objects” is a pre-processing step that is done before the encoding and is not a standardization issue [2]. A considerable amount of research has been and will be done on the segmentation of a scene into its “objects”. A *video object* (VO) is defined as spatio-temporal data associated with a semantically meaningful part of a natural scene, such as natural/synthetic human faces and bodies, scrolling text, overlaid graphics and sound [4, 5]. A *video object plane* (VOP) is a 2-D snapshot of a VO at a particular time instant.

The task of extracting (segmenting) objects from a video sequence is an important high-level video processing problem for object-based video manipulation and representation that has attracted many researchers. This task involves utilization of many low-level pre-processing tasks such as image segmentation, edge extraction and motion estimation. It is also very important to be able to assess the success of video object segmentation/tracking algorithms quantitatively and objectively in a correlated way with subjective experience.

This thesis aims first to develop and assess objective video segmentation performance measures. The second aim is to introduce a new algorithm for video object tracking in the light of the performance measures. In the course of the thesis work, some new low-level image and video processing algorithms have also been developed, which are an illumination invariant fuzzy image segmentation algorithm and a motion estimation algorithm in the frequency domain using fuzzy c-planes clustering.

1.2. Contribution and Scope

In the rest of this chapter, a taxonomy (classified survey) of video object segmentation and tracking algorithms is given. Video object segmentation is a high-level video processing task that involves many low-level processing steps such as image segmentation and motion estimation. Therefore, a brief survey of related image segmentation techniques existing in the literature is also provided.

Chapter 2 introduces two low-level image and video processing tasks which utilize the fuzzy clustering approach. In the first part of Chapter 2, an illumination invariant color image segmentation algorithm is presented, which makes use of the fuzzy c-planes algorithm for modeling illumination variations. In the second part of Chapter 3, a novel motion estimation algorithm in the frequency domain using fuzzy c-planes clustering is introduced.

Although there are many algorithms in the literature for video object segmentation and tracking, there are only a few studies to assess the goodness of the results. In Chapter 3, novel performance evaluation measures for object based video segmentation is introduced both when the ground-truth segmentation maps are available and without the existence of ground-truth maps.

In Chapter 4, a new video object tracking algorithm using performance evaluation measures as feedback is introduced. The algorithm consists of the *boundary prediction* and *boundary correction* stages. The boundary prediction stage aims to provide a coarse location of the object boundary using feature point tracking. The boundary correction stage deals with enhancing the accuracy of the tracked contour (possibly for non real-time applications) by using a closed-loop energy-minimization scheme. The feedback loop is used to locally adjust the algorithm parameters (energy weights) to obtain the optimal result for each segment of the object boundary. It is shown that this local adaptation strategy is superior to previous adhoc, fixed weight approaches.

Finally, in Chapter 5 concluding remarks and future research directions are given.

The contributions of this thesis can be itemized as follows:

1. A classified survey of existing video object segmentation and tracking, and still image segmentation algorithms are given.
2. A novel illumination invariant image segmentation algorithm for gray scale and color images is introduced.
3. A new motion estimation algorithm in the frequency domain using fuzzy c-planes clustering is developed.
4. Performance evaluation measures for object-based video segmentation both in the presence of ground-truth segmentation maps and without segmentation maps are introduced on a comparative basis.
5. An object tracking algorithm using the performance evaluation metrics in a feedback loop to locally adjust algorithm parameters is introduced, which outperforms previous fixed-weight approaches.
6. New energy terms using color and color-refined motion segmentation boundaries are introduced which are inherently normalized and therefore of comparable magnitude.

1.3. Literature Review

In this section, we give a taxonomy of video object segmentation and tracking algorithms existing in the literature. The problem of video object segmentation may involve utilization of many pre-processing techniques, such as color image segmentation. Therefore, a brief review of these techniques is also provided.

1.3.1. Video Object Segmentation and Tracking Techniques

Object-based video segmentation techniques aim to segment the moving objects in a scene for content-based processing and manipulation. Today, in movie and television production, techniques like blue screening (chroma keying) are used to capture and post-process video objects. It is also commonplace to produce natural video clips containing synthetic/natural objects with known alpha planes (segmentation maps),

such as logos, animated characters, overlaid texts [5]. Some approaches for automatic segmentation assume that the background is stationary (or has global motion that can be compensated using a parametric model) and the object of interest is moving independently of background motion. In this case, “semantic” object segmentation can be achieved simply by change detection and/or motion segmentation [6].

Extracting semantically meaningful objects using fully automatic methods is a difficult task. This is due to the fact that a semantic video object is not necessarily homogeneous with respect to color, motion or shape. Therefore, there are methods that propose effective use of user interaction to aid the process of object-extraction. Hence we classify the object-based video segmentation techniques as fully automatic and semi-automatic methods. In fact, the fully automatic techniques usually have some thresholds within the algorithms that are set by the user, which may be considered as a hidden (implicit) user interaction. However, in our classification user-interaction means explicit guidance provided by the user.

Some of the selected fully automatic and semi-automatic object tracking methods that will be described in the following sections are summarized in Table 1.1. In this table, we summarize the algorithms in terms of the assumptions and training they use, their user interaction requirements, the shape model used, the main method utilized and their intended task. While some of these methods emphasize real-time tracking of approximate object boundaries, others address pixel accurate object tracking in off-line mode. However, none is scalable to address both requirements in a single generic framework. The last row of this table belongs to the tracking framework introduced in Chapter 4 of this thesis, which aims to meet both requirements.

1.3.1.1. Fully Automatic Techniques. A method for layered representation of video was proposed by Wang and Adelson in [7]. In this work, each frame of a video sequence is decomposed into three layers: the intensity (texture) map, the alpha (segmentation) map, the velocity map. The automatic motion segmentation is carried out using an *adaptive k-means clustering* of the *affine motion* parameter vectors which are estimated

over non-overlapping square-blocks of the image. Then, each pixel of the image is assigned to one of the classes. A modification to this algorithm has been given in [8], which improves the k-means clustering stage.

Motion estimation and segmentation are inter-related in the sense that good motion estimation requires the motion segmentation boundaries and good motion segmentation requires accurate motion estimation. Therefore, there are methods that try to perform motion segmentation and motion estimation simultaneously [9]. In this work, a Bayesian framework is used that unifies motion estimation and motion segmentation.

Other methods impose constraints on the shape of the tracked object by using 2D [10, 11] or 3D shape [12, 13] models. Some of these algorithms acquire the 2D shape space information through training [14, 15], and use projections onto the shape space to estimate the most likely object boundary at a certain frame.

The CONDENSATION (conditional density propagation) algorithm [16], which is a state-space sampling approach, needs the shape space to be known beforehand. The shape space is constructed using B-spline curves and tracking is performed using probabilistic state-space sampling. Another application of learned motion models for tracking is presented in [17].

Pfinder (person finder) [18] is a blob tracking method, that runs in real time assuming that the background is relatively stationary. This method tracks human body parts, mainly the head and the hands using a 2D multiclass statistical model of color and shape. First, the background is learned without any person in the scene. Then, an entering person is detected and a model is constructed for tracking.

W^4 [19] (What, Where, When, Who) is an algorithm for real-time surveillance of people and their activities. It uses gray-scale monocular video to locate people and their body parts (head, hands, feet, torso) using mainly shape information. The background model is learned and updated in time.

An automatic method which is based on region merging is proposed in [20]. Given a set of initial regions, this method computes a spatio-temporal similarity between the regions and represents the regions and their pairwise symmetry values on a directed graph. The region merging is done using a two-stage graph-theoretic approach.

A segmentation algorithm which selects all the thresholds automatically is [21]. The segmentation algorithm distinguishes between stationary background, uncovered image regions, image regions to be covered and moving objects. The algorithm first uses a change detector and estimates the parametric motion iteratively. Then, the regions are labeled as above and finally the motion compensated prediction is carried out.

An efficient moving object segmentation algorithm is given in [22]. This block-based approach first extracts the background by observing several frames and collecting the blocks where there is no activity as background. Then, in each frame blocks different from the background are marked as foreground. Finally, the shape of the object is refined through elimination of small regions and block based low-pass filtering. A percentage error criterion with respect to the given alpha map is also presented.

A recent paper [23] introducing the idea of region binding is based on the idea that a compound region is represented and characterized by its unit regions. Therefore, the similarity between two compound regions is calculated using the similarity of their unit regions. It utilizes the watershed transform [24] and the moving object mask [25] to produce the moving object segmentation mask.

In [26], “good” feature points consisting of small square blocks are tracked throughout the image sequence. In [26], the goodness of a feature point is evaluated based on how well it can be tracked, which mainly utilizes the texture variation inside the feature block. To summarize very briefly, the “good” feature point locations are chosen such

that the minimum eigenvalue of the matrix:

$$\begin{bmatrix} \sum_{x \in FB} g_x^2 & \sum_{x,y \in FB} g_x g_y \\ \sum_{x,y \in FB} g_x g_y & \sum_{y \in FB} g_y^2 \end{bmatrix}, \quad (1.1)$$

is larger than a certain threshold. In the above equation, g_x and g_y denote the image gradients in the x and y directions and FB denotes the feature block (chosen as 7×7 during the experiments). The tracing of ‘good’ feature points is based on minimization of the motion compensated frame differences over feature blocks:

$$MSE = \sum_{FB} [f_2(\mathbf{x} + \mathbf{v}) - f_1(\mathbf{x})]^2, \quad (1.2)$$

where ϵ denotes the motion compensated block difference, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ denote two successive frames, \mathbf{x} denotes the x and y coordinates, and \mathbf{v} denotes the vector of motion parameters in the x and y directions, respectively. The similarity of the feature points is determined based on the above mean squared error (MSE) between the feature blocks. A translational (or affine) motion model is utilized.

Three essentially different fully automatic approaches for video segmentation will be evaluated using the performance evaluation measures which will be described in Section 3.1. The three approaches are: an edge-based [27], a motion clustering based [28], and a total feature vector clustering based [29] algorithm. These methods have been implemented and are described in detail below. As a by-product, improvements for two of these video segmentation algorithms are also presented.

These three approaches are selected specifically as they differ fundamentally in their algorithmic details. In the following we briefly describe the three selected video segmentation approaches via their simplified flowcharts and point out our modifications. The first two methods are fully automatic segmentation methods and the last one is a semi-automatic method.

I. Parametric Motion Segmentation Using Color Information: The first

video segmentation approach uses both the estimated dense motion field and the color segmentation for a region based parametric motion segmentation [28] to generalize the block based approach of [7]. The simplified flowchart of the algorithm is shown in 1.1. First, motion estimation is carried out between the current frame and the next frame using hierarchical Lucas-Kanade method. Then affine motion clusters are generated by classifying the dense motion field. Then, the current frame is color segmented using the fuzzy c-means algorithm. Finally, each color segment is assigned to one of the affine motion clusters. The intuitive approach behind this method is that, inaccurate motion boundaries are made more precise using the color segmentation boundaries.

The improvements brought to the algorithm are shown with shaded boxes:

- (a) One drawback of the Lucas-Kanade motion estimation algorithm is that, since it uses blocks around pixels to estimate the motion of the current pixel, the resulting motion vectors may sit astride on the object boundary. In order to mitigate this effect, we impose the color boundary constraint in the block around the current pixel, in other words only if the central pixel shares the same color segmentation label with the non-central pixel then its contribution is taken into account.
- (b) The number of motion classes is chosen automatically using cluster validity techniques [30, 31].
- (c) In order to make the color segmentation using the fuzzy c-means algorithm insensitive to minor local intensity variations, we used a local smoothing technique. During the calculation of the membership of a pixel, we look at the average of the memberships of the neighbors and perturb the membership of the current pixel towards this average, if the difference is small.

II. Binary Model Matching Approach Using Thresholded Frame Differences: An automatic moving object segmentation method by Meier et. al. [27] is based on matching a binary model of the object in the edge map of subsequent frames. The binary object model is extracted by thresholding the absolute difference of the first two frames.

The simplified flowchart of the Meier-Ngan approach [27] is given in 1.2, together with our proposed improvements, which are shown with shaded boxes:

- (a) In order to find the binary model of the object we use automatic thresholding of the difference of the first two frames instead of manual threshold selection as in [27]. Among the many algorithms that are tested, Otsu thresholding [32] gave more satisfactory results.
- (b) The original algorithm was designed for segmenting a single moving object [27]. In order to extend it to multiple objects, we propose two methods. The first one is based on partitioning the thresholded difference of the first two frames into square blocks. The number of edge pixels in each block is counted and the block is labeled as an object block if this number exceeds a certain percentage of the block size. After this, we apply connected components labeling to the labeled blocks in order to find out the actual number of objects in the scene and their approximate segmentation masks. Finally, each object is extracted and processed separately.

In the second approach for multiple object tracking, the Chamfer 3-4 distance transform of the binary frame difference image is thresholded so as to label just the pixels that are close to the object boundaries. Next, the number of objects and their approximate segmentation maps are extracted using connected component labeling.

III Interactive Segmentation Using Fuzzy C-means Clustering of Pixel

Feature Vectors: The third method that is analyzed [29], uses constrained fuzzy c-means clustering of a feature vector of each pixel. The feature vector of a pixel contains the Y, U, V color components, pixel location, texture calculated as the variance on a 3x3 window, and the optical flow vector. Each feature is normalized according to its range and weighted regarding the reliability of the motion estimation, e.g., when motion vectors are unreliable, then more weight is assigned to texture or color. The user interacts to combine the segmented regions to form the final objects. The flowchart of this algorithm is given in Figure 1.3.

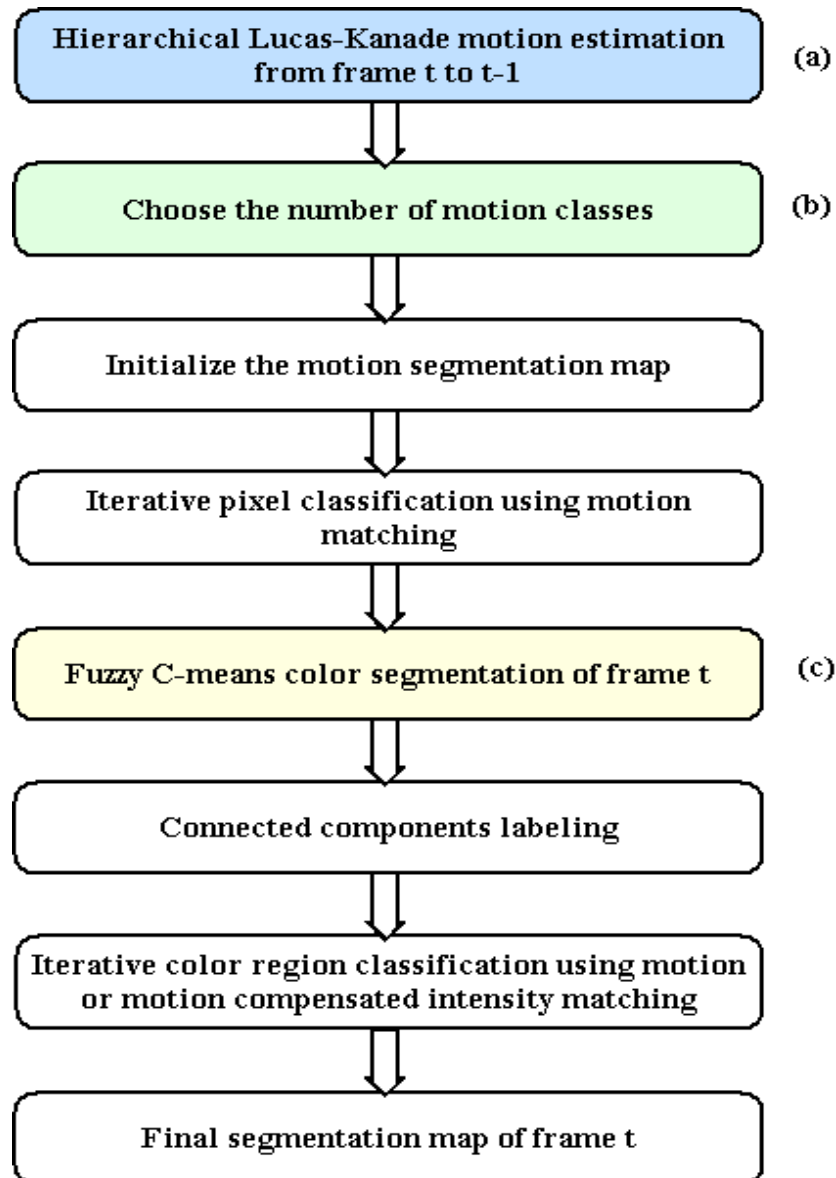


Figure 1.1. The simplified flowchart of the Altunbaşak-Eren-Tekalp method with the improved steps shown with shaded boxes

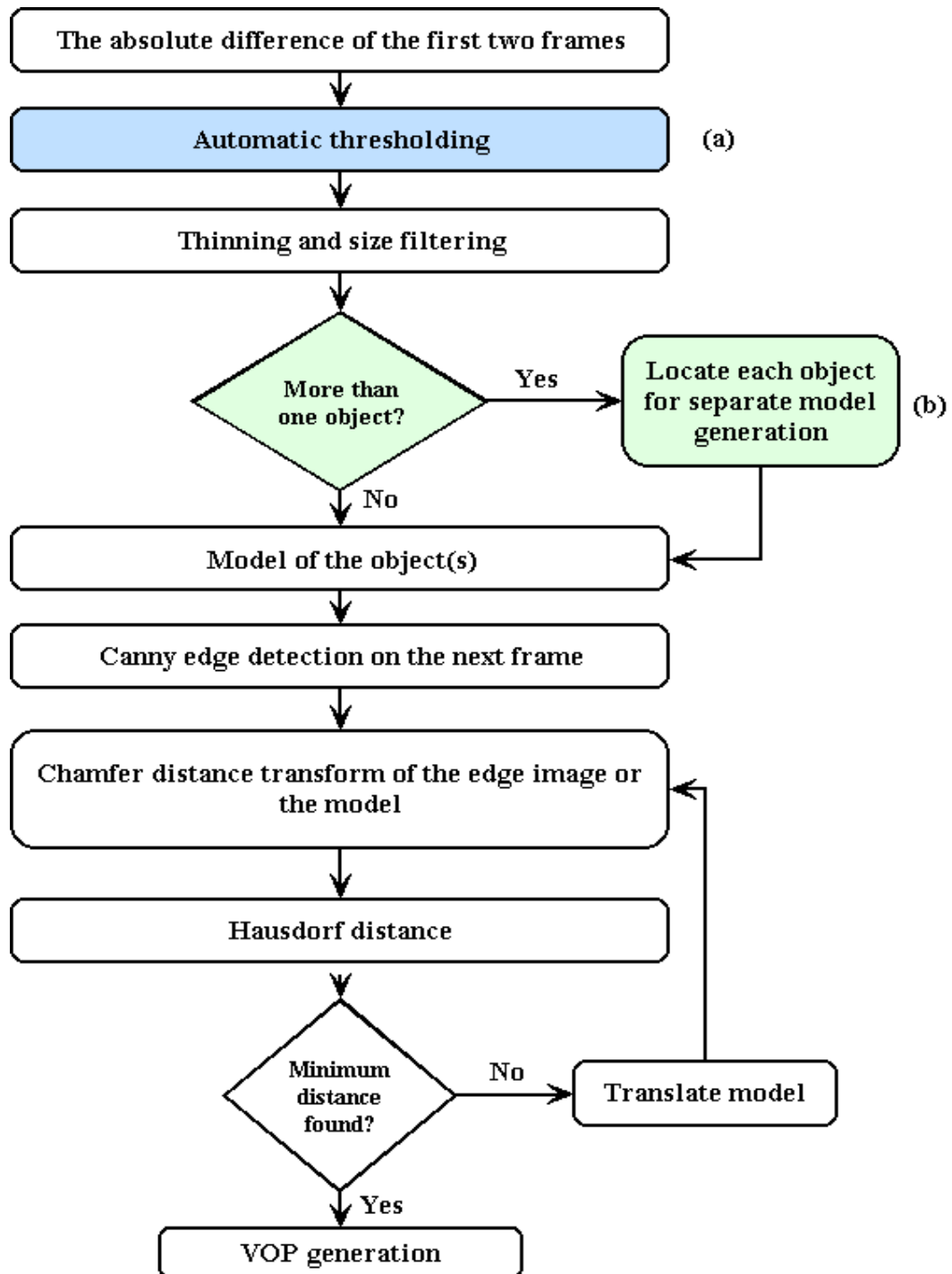


Figure 1.2. The simplified flowchart of the Meier-Ngan method with the improved steps shown with shaded boxes

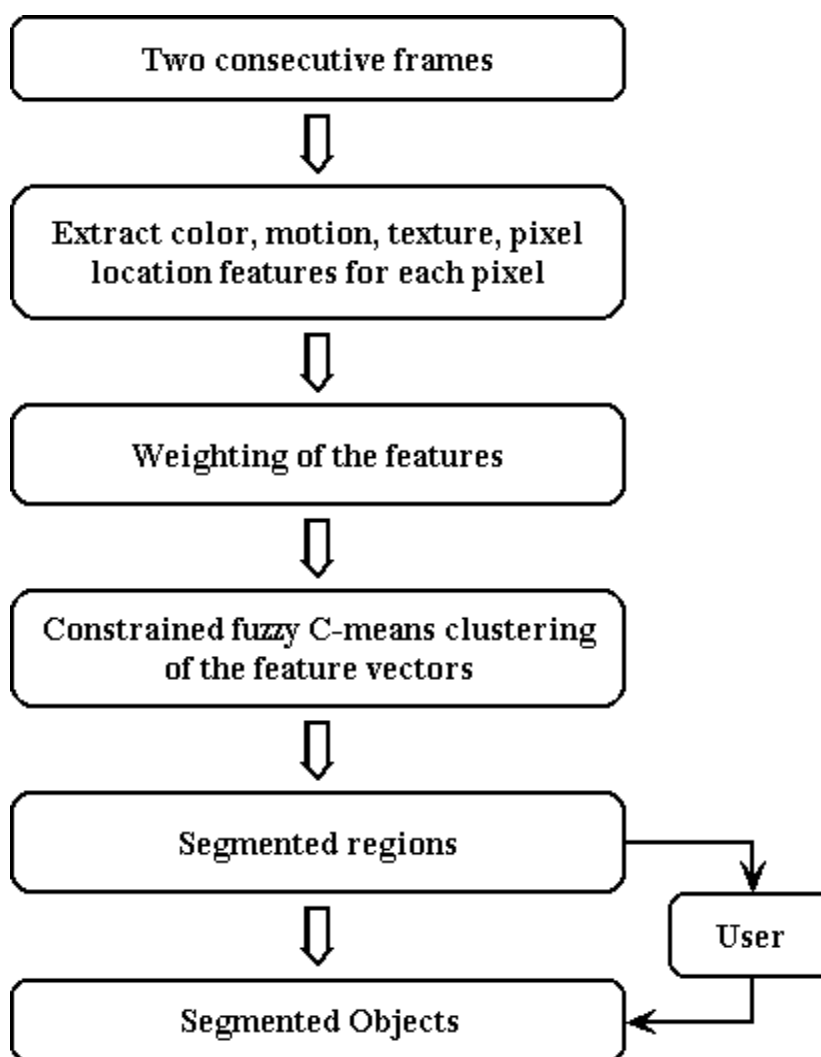


Figure 1.3. The simplified flowchart of the Castagno-Ebrahimi-Kunt method

Table 1.1. Some of the video object segmentation/tracking methods in the literature

Algorithm by	Assumptions, Training, Important Features	User Interaction	Shape Model	Method Used	Specific Task
Isard <i>et al.</i> [16] (Condensation)	Learn shape space using B-splines, state space sampling	None	Shape space using B-splines	State-space sampling	Track objects of shape space
Wren <i>et al.</i> [18] (Pfinder)	Single person, fixed camera, learn stationary background	None	Human body model	Blob Tracking	Human tracking
Wachter <i>et al.</i> [13]	Constant velocity, integrate edge and region information	Interactive initialization	Elliptical cones connected by joints	2D projection of the 3D model	Human tracking
Fu <i>et al.</i> [40]	Occlusion detection	Draw initial contour	None	Scale invariant snake	Track occlusion boundaries
Ju <i>et al.</i> [11] (Cardboard people)	Chain of patches	Define initial patches	Planar limbs	Predict location of next patch	Track human limbs
Cham <i>et al.</i> [10]	High dimensional state space, state-space search	User defines initial model	Scaled prismatic model	Multiple hypothesis testing	Articulated objects
Comaniciu <i>et al.</i> [33]	Uses color histogram, mean-shift analysis	Initial region marked	None (elliptical region)	Histogram matching using mean shift	Track regions of non-rigid objects
Baumberg <i>et al.</i> [15]	Uses B-splines, uniform motion, learn shape space automatically	None	Eigen-shapes	Kalman filter	Track walking humans
Haritaoglu <i>et al.</i> (W^4) [19]	Uses gray-scale images, learn background, learn projection templates	None	Silhouette analysis of people	Shape analysis and tracking	Surveillance of people
Altunbaşak <i>et al.</i> [28]	Color refined motion segmentation	None	None	Affine motion clustering	Segment moving objects
Meier <i>et al.</i> [27]	Binary frame difference matching, background stationary	None	Learn binary shape model	Hausdorff transform	Track moving objects
Castagno <i>et al.</i> [29]	Total feature vector clustering	User combines final regions	None	Fuzzy clustering	Track moving objects
Proposed Method	Performance feedback, new energy terms from color & motion segmentation boundaries	User draws initial boundary	None	Feature tracking, adaptive snake	Rigid and non-rigid objects

1.3.1.2. Semi-Automatic (User Interactive) Techniques. Semiautomatic segmentation and tracking approaches incorporate minimal user interaction to resolve the ambiguity of semantic meaning. The type of user interaction can be classified as *contour-based* and *region-based* interactions:

Contour-based Interaction: The boundary of the objects is outlined or marked in some of the frames in contour-based interactive algorithms. The automatic tracking algorithm tracks the object in subsequent frames with or without improving the accuracy of the boundary outline.

A real-time blob tracking approach that uses color histogram of the tracked object is presented in [33]. The boundary of the region to be tracked is marked by the user using an elliptical model. The histogram in the marked regions is tracked using mean shift analysis.

In *Mesh-based* segmentation methods [34, 5], the object boundaries are marked by user interaction in a small number of key frames. Then, objects are tracked automatically in subsequent frames using an occlusion adaptive content-based mesh representation. The object is tracked by estimating the motion of each node points in the mesh. An extension of this method for video object tracking using multiple key frames is given in [35]. This algorithm utilizes forward and backward mesh tracking and eliminates the need of keeping track of the uncovered regions as in [36, 34, 5].

Another semi-automatic object segmentation system [37] segments the I-frames interactively and the P-frames automatically. The user roughly marks the interior of the object in the I-frame after which a morphological watershed based algorithm snaps to the exact boundary. It is assumed that there is mild object motion between frames. The P-frame tracking is done using a 2D planar perspective transformation.

Algorithms that utilize the active contours for object tracking also require boundary-based interaction [38, 39]. The user specifies the contour of the object in the first frame in [40] and the object under occlusion is tracked in subsequent frames automatically.

In [39] the boundary of the VO is specified by the user on multiple frames and the object boundaries on intermediate frames using a boundary interpolation approach.

The work in [41] is based on tracking the moving objects based on deformable templates. The edges of the images (data features) are classified into clusters using the fuzzy c-means algorithm to find the centroids (data points) that will be associated to the model points for motion tracking. A Kalman filter based algorithm is used for tracking.

Region-based Interaction: In region-based algorithms, the user interacts with the regions that constitute the object. The user splits, merges, groups different segmented regions to arrive at a semantically meaningful object.

A recently proposed method uses multiple features of each pixel simultaneously [29]. The user interaction is at the end of the algorithm as opposed to the mesh based approach. The initial regions are extracted automatically and then the user groups these regions into semantically meaningful objects. The automatic region extraction is done using a fuzzy c-means clustering of the feature vectors of the pixels consisting of the three color components, displacement values, position values, texture information.

A region segmentation and tracking method based on color quantization and affine motion modeling is given in [42]. For intra-segmentation color is used and motion is used to track the color segments in time. Separate tracked regions are either grouped according to motion similarity or by user interaction.

In [43], an interactive segmentation based on a nested representation of the segmentation is presented. Using a modified watershed algorithm and a minimum spanning tree representation derived from the watershed algorithm, the segmentation is carried out at different partition levels. The user can interact with the program either locally or globally by selecting the final number of regions in the segmentation, or by clicking a certain part of the segmented image to refine the segmentation of that part by requiring more partitioning. Another type of interaction in [44, 43] is marker drawing.

User can draw different markers (lines, symbols) inside the objects and background to specify them. The exact location and shape of the markers are not very important. A temporal segmentation approach in [44] is based on change detection mask after camera motion compensation and scene cut detection.

Another approach of region tracking uses level sets [45, 46], in which the regions to be tracked are marked by the user in the first frame. The boundary of the region is implicitly modeled by the zero level set of a smooth function. The main advantage of level set-based methods over snake-based algorithms is that, it allows the boundary of the region to split and merge during tracking, providing topology independence.

1.3.2. Still Image Segmentation Techniques

We can define image segmentation as the process of dividing an image into non-overlapping regions that are uniform with respect to some criteria such as color and texture. The segmented regions should violate one or more of the uniformity conditions when they are merged. That implies there should be enough contrast between regions. There can also be higher level goals such as object extraction from the scene based on segmentation results. Although there are hundreds of algorithms for image segmentation, none of them can be considered as good for all kinds of images [47] and for all tasks. Several review papers exist on image segmentation [48, 49, 50, 47, 51]. These previous reviews do not adequately cover the new methods such as mesh-based based segmentation techniques, segmentation algorithms using genetic algorithms and hidden Markov models. Similarly, approaches on objective evaluation of segmentation outcomes are also not sufficiently elaborated. Therefore, in this section, a brief survey on image segmentation techniques including the above items is given.

Following the taxonomy given in [51], we will roughly classify previous work on image segmentation as **pixel-based** and **area-based** algorithms and methods for objective evaluation of image segmentation. Each of the above classes will be further subdivided.

A class of **pixel-based segmentation techniques** uses *thresholding methods*. Thresholding techniques are usually based on thresholding grey-level or color histograms of the images. For gray-scale images, the gray level histogram is used while for color images, color histogram is calculated in various ways using different color spaces. The color histograms are also calculated as 1D or 2D histograms. A method that fuses 2-D histograms of a color image is given in [52]. The thresholding method may be global or local. Global techniques use a single threshold for the entire image while local techniques calculate the threshold adaptively on sliding blocks of pixels. The thresholding methods can also be classified as bilevel thresholding or multithresholding [47]. There are various methods for automatically selecting the threshold value [32, 53]. Other approaches of pixel-based segmentation is use fuzzy clustering [31, 54, 55, 56] or its variations. A method that uses illumination invariant k-means clustering is given in [57].

Among the **area based segmentation techniques**, *region growing* and *split-merge* methods play an important role. There are also methods that use *morphology* [58, 59, 24]. A technique using the *mesh representation* is given in [60]. The approach is based on a hierarchical-mesh framework and the mesh representation around the object boundaries are processed. The user interacts with the system to put the object of interest in a bounding box. There also exist *Bayesian (MRF based) methods* [61]. A recent study [62] introduces a unified approach of region-based and edge-based features. The first and second order derivatives are incorporated into the Gibbs Random field (GRF) formulation in an attempt to prevent under-segmentation due to excessive smoothing (hence uses first order derivatives) and to impose stronger spatial homogeneity constraints within coherent color regions (hence uses second order derivatives). There are a few recent approaches that utilize the *genetic algorithms* [63, 64, 65]. In [64], a MSE criterion using an MRF model of the regions is minimized via genetic algorithm since the criterion can not be easily minimized with usual optimization methods. Another approach of segmentation is based on *hidden markov model based methods*. For example, the method in [66] utilizes the two-dimensional hidden markov model in order to build a context dependent classifier by exploiting the inter-block dependency of the images. An application of this approach on classification of document images have

been shown to successfully segment text and pictures in a given document [66]. One drawback of the HMM based method is that it needs to be trained beforehand. There are also *neural network based* approaches [67, 52].

There are also methods that unify **edge-based segmentation techniques** with region-based approaches [68, 69, 70, 71]. **Object (physics) based segmentation techniques** try to model the illumination and the material changes in the image by using physical models [72, 73, 74].

There are efforts for **objective evaluation of image segmentation** [75, 76, 77, 78, 79, 80]. Quantifying the performance of a segmentation algorithm is important in terms of providing a feedback to the segmentation system [77]. In [77], the quality of segmentation is measured using uniformity within segmented regions and contrast between adjacent regions. The survey paper [81] classifies evaluation criteria as goodness based on intra-region uniformity, inter-region contrast, region and shape; discrepancy based on mis-segmented pixels, the position of the mis-segmented pixels, the number of objects in the scene, the feature values of segmented objects. A comprehensive survey is also given in [52]. In [82], threshold selection techniques are evaluated based on a busyness criterion and an error criterion.

2. LOW-LEVEL IMAGE AND VIDEO PROCESSING TECHNIQUES USING FUZZY PROCESSING

This chapter introduces two novel low-level image and video processing techniques using fuzzy clustering [83] developed in the context of image segmentation and motion estimation. In Section 2.1, an illumination invariant fuzzy image segmentation algorithm will be introduced. In Section 2.2, a motion estimation algorithm in the frequency domain using fuzzy c-planes clustering will be presented.

2.1. Illumination Invariant Fuzzy Image Segmentation

Local illumination variations may exist in images causing a handicap in the segmentation task. For example, MR images may have different intensities in their different parts. Strong illumination variations and shading effects also occur in natural scenes. Previous work that addresses the problem of illumination variations in segmentation of MR images can be found in [84, 85, 86, 87]. A recent study [56] proposes an adaptive fuzzy c-means algorithm for image segmentation in the presence of intensity inhomogeneities. It uses a multiplier field to model the intensity inhomogeneities which need to be solved for iteratively.

In this section, we present a novel algorithm to segment images in the presence of local illumination variations [88]. For this purpose, first we introduce an image segmentation algorithm using fuzzy c-planes (FCP) [83] clustering, where the mean intensity value used in the classical fuzzy c-means algorithm [89, 90] is replaced with a plane fitting approach to take into account the spatial illumination variations. The method we present may be extended to the more general “fuzzy c-surfaces” scheme by fitting surfaces of higher order instead of planes. Secondly, the method includes a local regularization scheme for the updating of the membership functions. The main motivation behind the local regularization is to make the membership of an image pixel closer to the mean of the memberships of its neighbors, but avoiding smoothing over large differences between the neighbors, which may be due to genuine segment

boundaries. As expected, the local regularization also helps to eliminate the effects of noise. Thirdly, in order to constrain the segmented regions to be spatially connected, we incorporate in the feature vector the weighted location information of the pixels along with the intensity values. The plane (or surface) fitting approach we present does not need any time consuming iterations and if prior information is available, it can be modified according to characteristics of the illumination variation throughout the image.

2.1.1. Image Segmentation Using Fuzzy c-planes Clustering

The objective function to be minimized in the fuzzy c-planes image segmentation is:

$$J_{FCP} = \sum_{i=1}^C \sum_{(x,y)} m_i^q(x,y) D_i^2(x,y), \quad (2.1)$$

where

$$D_i(x,y) = \left\| \begin{bmatrix} f(x,y) \\ \alpha x \\ \alpha y \end{bmatrix} - \begin{bmatrix} a_i x + b_i y + d_i \\ \alpha \bar{x}_i \\ \alpha \bar{y}_i \end{bmatrix} \right\|, \quad (2.2)$$

and C denotes the total number of classes, $m_i(x,y)$ denotes the membership value at pixel location (x,y) for class i such that $\sum_{i=1}^C m_i(x,y) = 1$; $f(x,y)$ denotes the intensity value of the pixel at location (x,y) ; a_i, b_i, d_i are the parameters of the plane fit to class i , (\bar{x}_i, \bar{y}_i) denote the location of the cluster centroid, α denotes the relative importance attached to the spatial coordinate information, and q is an exponent parameter controlling the fuzziness of the classification. The larger q is, the fuzzier the classification becomes. The operator $\|\cdot\|$ denotes any distance function which we in this study chose as the Euclidean distance.

Recall that in the classical fuzzy c-means clustering based image segmentation,

the pixel intensity $f(x, y)$ is compared with the mean intensity of each class, that is:

$$\bar{f}_i = \frac{\sum_{(x,y)} m_i^q(x, y) f(x, y)}{\sum_{(x,y)} m_i^q(x, y)}. \quad (2.3)$$

Instead, we assume a first order polynomial trend of the segment intensities modeled as $ax + by + d$.

The steps of the algorithm proposed for the minimization of the above objective function are:

I. Choose C initial cluster centroids as:

$$\mathbf{c}_i = \begin{bmatrix} \bar{f}_i \\ \alpha \bar{x}_i \\ \alpha \bar{y}_i \end{bmatrix}, i = 1, \dots, C. \quad (2.4)$$

where \bar{f}_i denotes the average gray level of the i^{th} cluster, (\bar{x}_i, \bar{y}_i) denote the location of the cluster centroid, and a_i, b_i in (2.2), which are the plane parameters, are chosen as zero. The weight parameter α adjusts the relative importance of the coordinates with respect to the intensity value. The initial average gray levels are chosen as the local peaks of the histogram of the image.

Set the iteration counter z to 1.

II. Compute the membership of pixel (x, y) to class i as:

$$m_i(x, y) = \frac{\|D_i(x, y)\|^{-2/(q-1)}}{\sum_{k=1}^C \|D_k(x, y)\|^{-2/(q-1)}}, \quad i = 1, \dots, C. \quad (2.5)$$

Increment iteration counter z by 1.

III. Smooth the membership values using the following local regularization:

$$\hat{m}_i(x, y) = m_i(x, y) + \mu(m_{i,av} - m_i(x, y)) \exp\left(-\frac{(m_{i,av} - m_i(x, y))^2}{\sigma^2}\right) \quad (2.6)$$

where

$$m_{i,av} = \frac{1}{N_\zeta} \sum_{(x,y) \in \zeta} m_i(x, y) \quad (2.7)$$

and ζ denotes a neighborhood (usually the eight neighbors) of pixel at location (x, y) , N_ζ denotes the number of pixels in the neighborhood. The parameters σ and μ adjust the sensitivity of the membership of the current pixel to its neighbors. Then let,

$$\tilde{m}_i(x, y) = \frac{\hat{m}_i(x, y)}{\sum_{k=1}^C \hat{m}_k(x, y)} \quad (2.8)$$

for normalization of the membership values so that they still add up to 1, and assign $m_i(x, y)$ to $\tilde{m}_i(x, y)$.

Our aim in performing the local regularization given in (2.6) is to adjust the membership value of the current pixel towards the average membership value of its neighbors if the difference between them is not large.

In (2.6), the variance of the Gaussian function can be adjusted in a decreasing manner as the iterations proceed. For example we may set:

$$\sigma = \frac{1}{z^n}, \quad (2.9)$$

where n is a positive integer and z is the iteration counter. As the variance of the Gaussian function decreases, the neighbors have less and finally no effect on the membership values of the current pixel. By this way, we do not disturb the convergence properties of the original fuzzy c-means algorithm, which still holds in our case.

IV. Find the plane parameters and the position of class centroids by minimizing:

$$\sum_{(x,y)} m_i^q(x, y) [(f(x, y) - (a_i x + b_i y + d_i))^2 + (\alpha x - \alpha \bar{x}_i)^2 + (\alpha y - \alpha \bar{y}_i)^2] \quad (2.10)$$

We can do this separation since for each i in (2.1) the corresponding summation

over all the pixels have a positive value. So, it suffices to minimize each individual term to minimize the overall expression. We can further split (2.10), to find plane parameters that minimizes:

$$\sum_{(x,y)} m_i^q(x, y)[f(x, y) - (a_i x + b_i y + d_i)]^2 \quad (2.11)$$

We take the partial derivatives of the above expression with respect to a_i , b_i and d_i and equate them to zero, which results in the following system of equations:

$$\begin{bmatrix} \sum m_i^q(x, y)x^2 & \sum m_i^q(x, y)xy & \sum m_i^q(x, y)x \\ \sum m_i^q(x, y)xy & \sum m_i^q(x, y)y^2 & \sum m_i^q(x, y)y \\ \sum m_i^q(x, y)x & \sum m_i^q(x, y)y & \sum m_i^q(x, y) \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ d_i \end{bmatrix} = \begin{bmatrix} \sum f(x, y)m_i^q(x, y)x \\ \sum f(x, y)m_i^q(x, y)y \\ \sum f(x, y)m_i^q(x, y) \end{bmatrix} \quad (2.12)$$

Find the location of the cluster centroids that minimizes the following expression:

$$\sum_{(x,y)} m_i^q(x, y)[(\alpha x - \alpha \bar{x}_i)^2 + (\alpha y - \alpha \bar{y}_i)^2]. \quad (2.13)$$

After equating the partial derivatives to zero,

$$\bar{x}_i = \frac{\sum_{(x,y)} m_i^q(x, y)x}{\sum_{(x,y)} m_i^q(x, y)} \quad (2.14)$$

$$\bar{y}_i = \frac{\sum_{(x,y)} m_i^q(x, y)y}{\sum_{(x,y)} m_i^q(x, y)} \quad (2.15)$$

- V. Iterate until the sum of the error between two successive values a_i , b_i and d_i is less than a threshold value.
- VI. Assign each pixel to the cluster of highest membership.

Instead of fitting a plane, we can also fit higher order surfaces to the data, to make the algorithm robust to other nonlinear intensity inhomogeneities.

The parameter α in Step I is chosen such that the relative weight of the intensity d_i to the x and y coordinates is around 4. The parameters μ and σ in Step III are chosen around the values 1 and 0.5, respectively. The threshold for the stopping criterion in Step V is chosen around 0.001.

If we start with a small number of classes and wish to increase the number of classes as necessary, an efficient cluster center selection scheme would be choosing a new class near an existing class with the largest fit error. The fit error for class i is defined as:

$$E_i = \sum_{(x,y)} \left\| \begin{bmatrix} f(x,y) \\ \alpha x \\ \alpha y \end{bmatrix} - \begin{bmatrix} a_i x + b_i y + d_i \\ \alpha \bar{x}_i \\ \alpha \bar{y}_i \end{bmatrix} \right\| \quad (2.16)$$

The fit error can also be used as a cluster validity criterion for the FCP algorithm. Other cluster validity criteria using the membership function [91, 92, 31] can also be used.

2.1.2. Extension to Color Images

The color space that is chosen for illumination invariant color image segmentation is the HSV (hue-saturation-value) color space. The V component is the one that carries the illumination information. So, we can fit a plane for the V component and represent H and S components with cluster means. The objective function to be minimized in the fuzzy c-planes color image segmentation is:

$$J_{FCP} = \sum_{i=1}^C \sum_{(x,y)} m_i^q(x,y) D_i^2(x,y) \quad (2.17)$$

where

$$D_i(x,y) = \left\| \begin{bmatrix} h(x,y) \\ s(x,y) \\ v(x,y) \\ \alpha x \\ \alpha y \end{bmatrix} - \begin{bmatrix} \bar{h}_i \\ \bar{s}_i \\ a_i x + b_i y + d_i \\ \alpha \bar{x}_i \\ \alpha \bar{y}_i \end{bmatrix} \right\| \quad (2.18)$$

and C denotes the total number of classes, $m_i(x, y)$ denotes the membership value at pixel location (x, y) for class i such that, $0 < m_i(x, y) < 1$; $h(x, y)$, $s(x, y)$ and $v(x, y)$ denote the HSV values of a pixel; \bar{h}_i, \bar{s}_i denote the i^{th} cluster centroid for hue and saturation and a_i, b_i, d_i are the parameters of the plane fit to class i of the v component and q is an exponent parameter controlling the fuzziness of the classification.

The steps of the algorithm proposed for the minimization of the above objective function are:

I. Choose C initial cluster centroids as:

$$\mathbf{c}_i = \begin{bmatrix} \bar{h}_i \\ \bar{s}_i \\ \bar{v}_i \\ \alpha \bar{x}_i \\ \alpha \bar{y}_i \end{bmatrix}, i = 1, \dots, C, \quad (2.19)$$

where \bar{h}, \bar{s} and \bar{v} denote the average hue, saturation and value components of the cluster, (m_i, n_i) denote the location of the cluster centroids, and a_i, b_i in (2.18), which are the plane parameters of the value component, are chosen as zero. The initial average H, S, V values are chosen as the local peaks of the color histogram of the image. In (1), α denotes the relative importance attached to the spatial coordinate information.

Set the iteration counter z to 1.

II. Compute the membership of pixel (x, y) to class i as:

$$m_i(x, y) = \frac{\|D_i(x, y)\|^{-2/(q-1)}}{\sum_{k=1}^C \|D_k(x, y)\|^{-2/(q-1)}}, \quad i = 1, \dots, C, \quad (2.20)$$

Increment the iteration counter z by 1.

III. Smooth the membership values using the following local regularization:

$$\hat{m}_i(x, y) = m_i(x, y) + \mu(m_{i,av} - m_i(x, y)) \exp\left(-\frac{(m_{i,av} - m_i(x, y))^2}{\sigma^2}\right) \quad (2.21)$$

where

$$m_{i,av} = \frac{1}{N_\zeta} \sum_{(x,y) \in \zeta} m_i(x,y) \quad (2.22)$$

and ζ denotes a neighborhood (usually the eight neighbors) of pixel at location (x, y) , N_ζ denotes the number of pixels in the neighborhood. The parameters σ and μ adjust the sensitivity of the current pixel to its neighbors. Then let,

$$\tilde{m}_i(x,y) = \frac{\hat{m}_i(x,y)}{\sum_{k=1}^C \hat{m}_k(x,y)} \quad (2.23)$$

for normalization of the membership values so that they still add up to 1, and assign $m_i(x,y)$ to $\tilde{m}_i(x,y)$.

Our aim in performing the local regularization given in (2.21) is to adjust the membership value of the current pixel towards the average membership value of its neighbors if the difference between them is not large.

In (2.21), the variance of the Gaussian function can be adjusted in a decreasing manner as the iterations proceed. For example we may set:

$$\sigma = \frac{1}{z^k}, \quad (2.24)$$

where k is a positive integer and z is the iteration counter. As the variance of the Gaussian function decreases, the neighbors have less and finally no effect on the membership values of the current pixel. By this way, we do not disturb the convergence properties of the original fuzzy c-means algorithm, which still holds in our case.

IV. Find the cluster centroids from

$$\bar{h}_i = \frac{\sum_{(x,y)} h(x,y) m_i^q(x,y)}{\sum_{(x,y)} m_i^q(x,y)} \quad (2.25)$$

$$\bar{s}_i = \frac{\sum_{(x,y)} s(x,y) m_i^q(x,y)}{\sum_{(x,y)} m_i^q(x,y)} \quad (2.26)$$

$$\begin{bmatrix} a_i \\ b_i \\ d_i \end{bmatrix} = \mathbf{A}^{-1}\mathbf{B} \quad (2.27)$$

where

$$\mathbf{A} = \begin{bmatrix} \sum_{(x,y)} m_i^q(x,y)x^2 & \sum_{(x,y)} m_i^q(x,y)xy & \sum_{(x,y)} m_i^q(x,y)x \\ \sum_{(x,y)} m_i^q(x,y)xy & \sum_{(x,y)} m_i^q(x,y)y^2 & \sum_{(x,y)} m_i^q(x,y)y \\ \sum_{(x,y)} m_i^q(x,y)x & \sum_{(x,y)} m_i^q(x,y)y & \sum_{(x,y)} m_i^q(x,y) \end{bmatrix} \quad (2.28)$$

$$\mathbf{B} = \begin{bmatrix} \sum_{(x,y)} v(x,y)m_i^q(x,y)x \\ \sum_{(x,y)} v(x,y)m_i^q(x,y)y \\ \sum_{(x,y)} v(x,y)m_i^q(x,y) \end{bmatrix} \quad (2.29)$$

V. Iterate until the sum of the error between two successive values a_i , b_i and d_i is less than a threshold value T .

VI. Assign each pixel to the cluster of highest membership.

2.1.3. Experimental Results

In order to test the presented image segmentation approach we first used a 30×30 image that had two rectangles of uniform intensity (gray levels 50 and 255), and two stripes with linear intensity variations (from 65 to 210), which is shown in Figure 2.1(a) and 2.2(a). There is also additive white Gaussian noise with peak signal to noise ratio (PSNR) of 10dB and 4dB, respectively. In Figure 2.1(b) and Figure 2.1(c) we give the segmentation results for four classes obtained using the standard fuzzy c-means algorithm (FCM) and the proposed fuzzy c-planes (FCP) algorithm, respectively.

We can see that the standard FCM results in an incorrect segmentation regarding the region boundaries, whereas the FCP algorithm can correctly identify the four regions of the original image. The final gray levels for the class centers are: [147, 172, 248, 7]. The results when we decrease the PSNR of the original image to 4dB are given in Figure 2.2.

Another example is given in Figure 2.3, where the original image consists of two balls illuminated with two different light sources at the top and at the bottom. The background consists of two parts that have an intensity gradient as shown in Figure 2.3(a), and the PSNR is chosen as 10dB. The standard FCM algorithm using four classes is not able to segment the background correctly in two parts as shown in Figure 2.3(b). This is both due to the noise and the intensity gradient of the background. The FCP algorithm segments the background and the ball on the left better than the FCM algorithm. FCP is more robust to noise and the intensity variations.

The first experiment using color images is performed on a synthetic image that is shown in Figure 2.4. The hue values of the regions are: 0 for red, 0.25 for green, 0.5 for cyan, 0.75 for dark blue. The saturation values of all the regions are chosen as 1. The value component of the top and bottom rectangles are chosen linearly varying in a range between 0 and 1. Also Gaussian noise is added to this component such that the PSNR is 4dB. As shown in Figure 2.4 (b), the FCM (fuzzy c-means) algorithm is not very successful, whereas the FCP (fuzzy c-planes) algorithm finds the region boundaries correctly. In Figure 2.5, the noise level is increased such that the PSNR is 0dB, where the FCP based algorithm still gives perfect results.

Another experiment using six classes is performed on the ROBOT image which is shown in Figure 2.6 (a). It can be seen that the fuzzy c-planes algorithm is more successful (see Figure 2.6 (c)) in segmenting the robot arm as one class, whereas FCM algorithm splits it into two (see 2.6 (b)). The red and yellow rectangular pieces are also incorrectly segmented with their shadows in Figure 2.6 (b), whereas the segmentation using the FCP method is more successful.

The results on the COLA CAN image is given in Figure 2.7 for four classes. The results of the FCM and FCP algorithms are similar except the upper left part of the can, where the illumination effects can be seen most and the FCP gives better segmentation results.

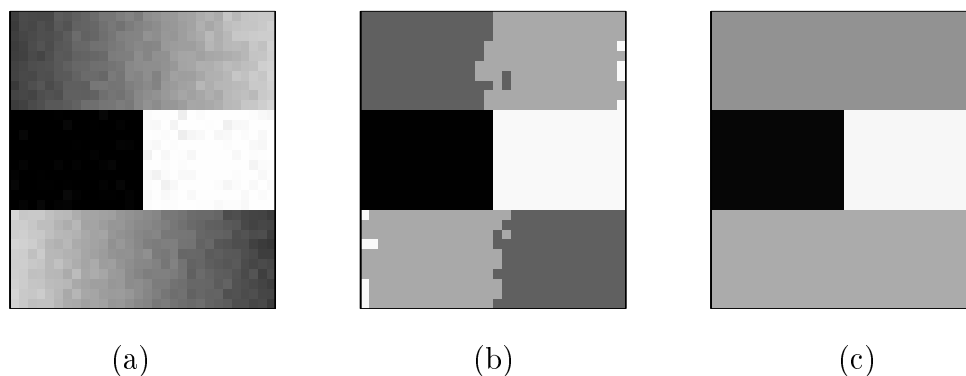


Figure 2.1. (a) The original image, PSNR = 10dB (b) The result using the FCM algorithm (c) The result using the proposed FCP algorithm, where gray levels for the class centers are: 147, 172, 248, 7

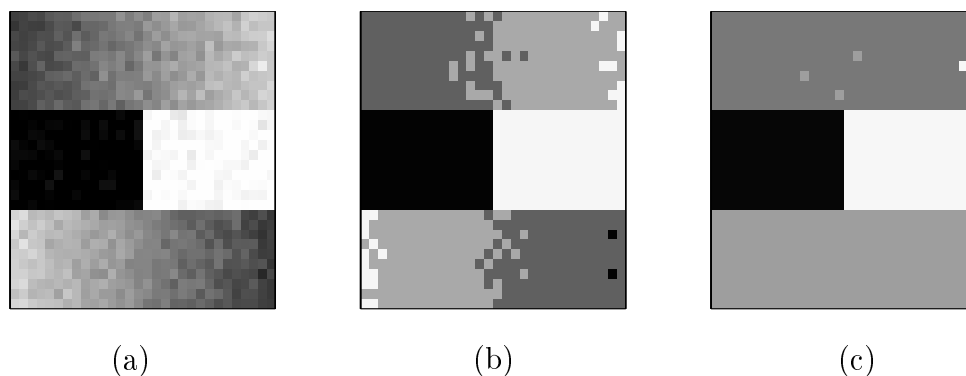


Figure 2.2. (a) The original image, PSNR = 4dB (b) The result using the FCM algorithm (c) The result using the proposed FCP algorithm

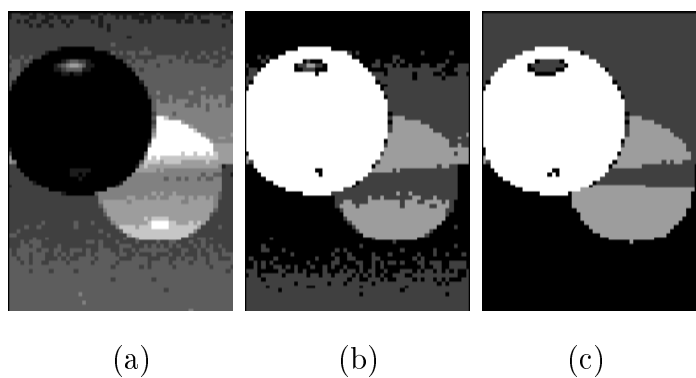


Figure 2.3. (a) The original image, PSNR = 10dB (b) The result using the FCM algorithm (c) The result using the proposed FCP algorithm

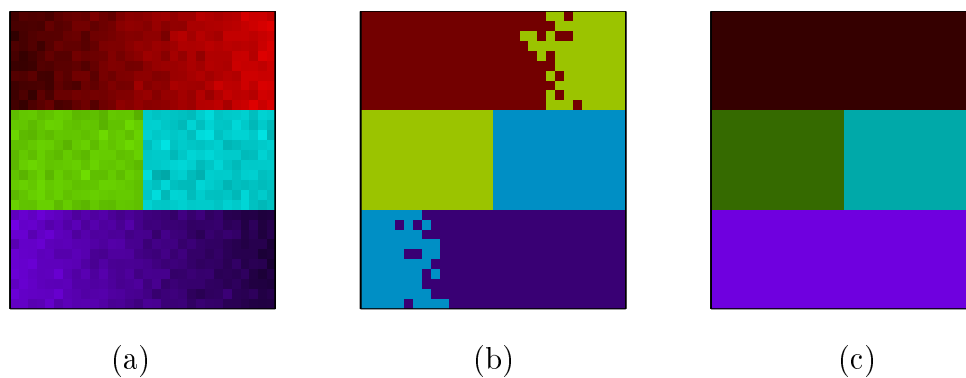


Figure 2.4. (a) Original image, PSNR = 4dB (b) Segmentation result of the fuzzy c-means algorithm (c) Segmentation result of the proposed fuzzy c-planes algorithm

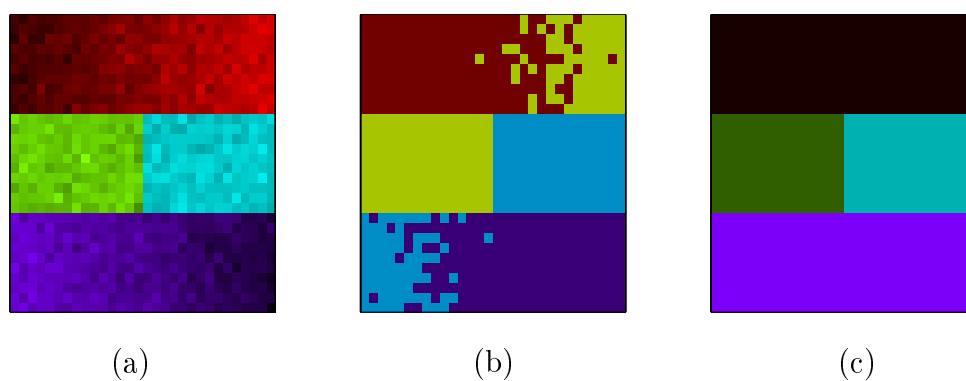


Figure 2.5. (a) Original image PSNR = 0dB (b) Segmentation result of the FCM algorithm (c) Segmentation result of the proposed FCP algorithm

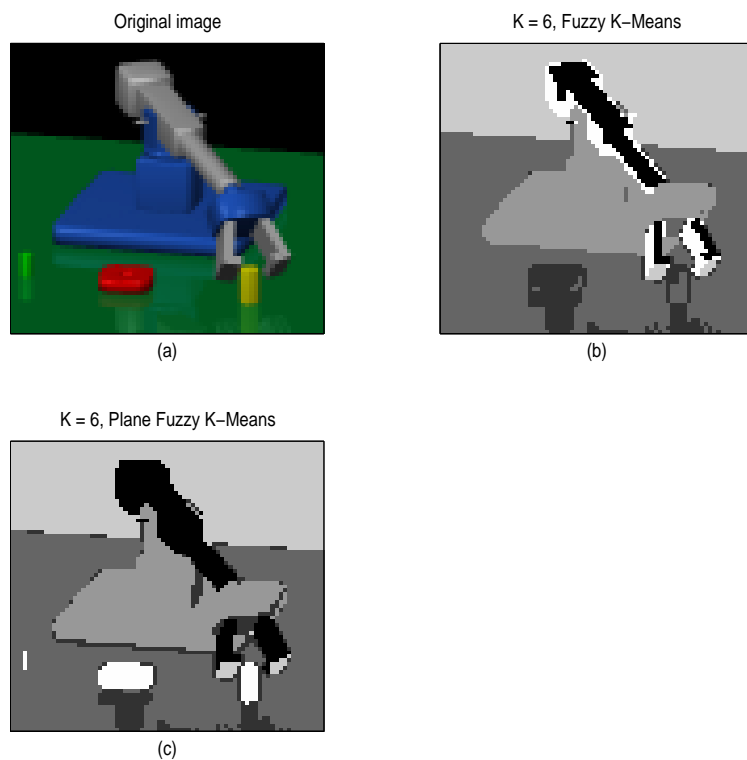


Figure 2.6. (a) The original ROBOT image (b) Segmentation result of the FCM algorithm (c) Segmentation result of the proposed FCP algorithm

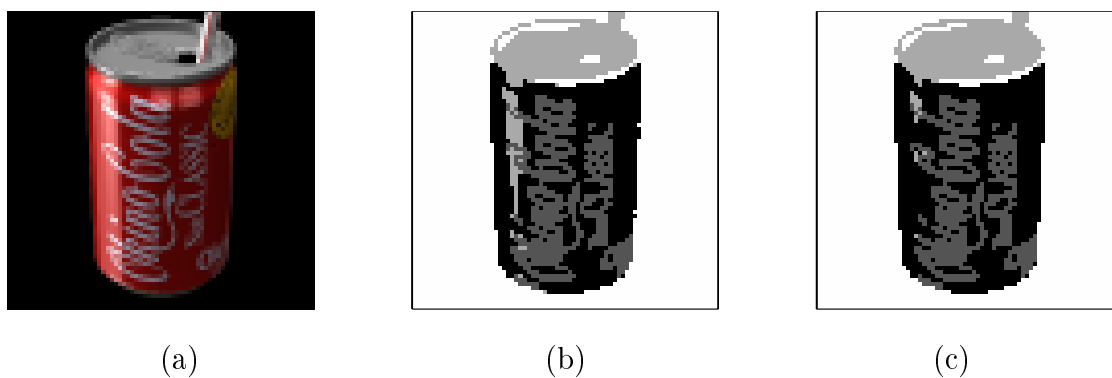


Figure 2.7. (a) The original COLA CAN image (b) Segmentation result of the FCM algorithm (c) Segmentation result of the proposed FCP algorithm

2.2. Motion Estimation in the Frequency Domain Using Fuzzy C-Planes Clustering

Motion estimation is an important problem in video processing that is utilized in many applications such as video coding, object-based video manipulation, object-based segmentation and dynamic scene analysis. Many studies have been carried out in an effort to estimate the motion of the objects in a scene [93, 94, 95, 96, 97, 98]. There are a number of approaches based on the correlation methods [94, 96] and the optical flow constraint [94, 95, 96]. These methods generally use regularization techniques assuming the smoothness of the motion field in order to obtain a unique solution. This smoothness assumption has an unwanted side effect of over-smoothing the motion discontinuities.

In a recent work [99, 93], the discontinuous motion estimation problem is handled in a harmonic retrieval framework. In this approach, the motion discontinuities are explicitly modeled and the velocity estimation is carried out independently of the shape of the moving object and the density of motion discontinuities. The moving objects are assumed to have different translational motion parameters and the motion is assumed to be time-invariant in a short time interval. In order to find the motion parameters, first, horizontal and vertical component sets along with their corresponding sums are estimated from the peak locations of periodogram analyses [99]. Periodogram analysis is a non-parametric method for spectrum estimation. A component pairing approach follows this step which is prone to errors. In this work, we present a more robust method for estimation of the motion parameters [100], which uses the 3-D frequency domain data and fits planes to them via the fuzzy c-planes clustering method [101, 102, 83]. This approach has been indicated in [99] as future work with no results given.

In Section 2.2.1, the theoretical background for modeling the motion estimation problem in the frequency domain [99] is introduced. In Section 2.2.2, the motion parameter estimation method using the fuzzy c-planes clustering approach is presented. In Section 2.2.3, the experimental results are given and compared with the pairing method of [99].

2.2.1. Theoretical Background

In this section, the multiple discontinuous motion modeling approach described in [99] is briefly summarized. The reader should refer to [99] for a detailed explanation of notation and equations.

Given an image sequence, it is assumed that there exists M (including the background) objects (partitions) in the scene at time t , i.e.,

$$W(t) = \bigcup_{k=1}^M W_k(t), \quad W_k(t) \cap W_n(t) = \emptyset, \quad \forall k \neq n, \quad (2.30)$$

where $W(t)$ denotes the region of interest of the image at time t , and $W_k(t)$ is the support region of object k . Let $f_{Ik}(x, y)$ denote the “ideal” image of object k , at time $t = 0$, if there is no occlusion. Let the parameters $(u_i, v_i), i = 1, \dots, M$, denote the motion vectors of the objects, which are assumed to be constant in a short time interval T . Then, the image sequence $f(x, y; t), t = 0, \dots, T$, can be represented as [99]:

$$f(x, y; t) = \sum_{k=1}^M f_{Ik}(x - u_k t, y - v_k t) I_k(x, y; t), \quad (2.31)$$

where $I_k(x, y; t)$ denotes the indicator function representing $W_k(t)$, which takes the value 1 if $(x, y) \in W_k(t)$, and takes the value 0 otherwise.

After some manipulations [99], (2.31) can be written as:

$$f(x, y; t) = \sum_{k=1}^M f_{vk}(x - u_k t, y - v_k t) + \varepsilon(x, y; t) \quad (2.32)$$

$$f_{vk}(x - u_k t, y - v_k t) = f_{Ik}(x - u_k t, y - v_k t) \bar{w}_{k0}(x - u_k t, y - v_k t) \quad (2.33)$$

where f_{Ik} is the ideal image of object k , and \bar{w}_{k0} denotes its corresponding average support window over T frames [99]:

$$\bar{w}_{k0}(x, y) = \frac{1}{T} \sum_{t=0}^T I_k(x + u_k t, y + v_k t; t). \quad (2.34)$$

The term,

$$\varepsilon(x, y; t) = \sum_{k=1}^N f_{Ik}(x - u_k t, y - v_k t) \xi_k(x, y; t) \quad (2.35)$$

denotes the error, and

$$\xi_k(x, y; t) = I_k(x, y; t) - \overline{w}_{k0}(x - u_k t, y - v_k t) \quad (2.36)$$

denotes the difference between the support window of object k at time t and its shifted average support window. The component $f_{vk}(x - u_k t, y - v_k t)$ can be thought as the portion of $f_{Ik}(x, y)$ which undergoes translational motion from frame to frame and the error term in (2.32) results from the occluded and uncovered regions of the object. The 2-D spatial Fourier transform of (2.32) gives:

$$F(\omega_x, \omega_y; t) = \sum_{k=1}^M F_{vk}(\omega_x, \omega_y) e^{-j(\omega_x u_k + \omega_y v_k)t} + E(\omega_x, \omega_y; t). \quad (2.37)$$

When the (ω_x, ω_y) pair is fixed, $F(\omega_x, \omega_y; t)$ is a 1-D signal consisting of M harmonics in noise where the frequency of the k^{th} harmonic is:

$$\omega_k = -\omega_x u_k - \omega_y v_k, \quad k = 1, \dots, M. \quad (2.38)$$

Our aim is to estimate the motion parameters $(u_k, v_k), k = 1, \dots, M$. Periodogram analysis is a classical method for estimating the parameters of harmonics in noise [103], and is based on the Fourier transform of the data sequence. Another method for frequency estimation is the multiple signal classification (MUSIC) algorithm [104], which can provide a higher resolution than the periodogram. For a specific (ω_x, ω_y) pair, the periodogram of $F(\omega_x, \omega_y; t)$ which is based on its temporal Fourier transform is given by:

$$P_F(\omega_x, \omega_y; \omega) = \frac{1}{T} \left| \sum_{t=0}^{T-1} F(\omega_x, \omega_y; t) e^{-j\omega t} \right|^2 \quad (2.39)$$

the peaks of which give the component frequencies $\omega = -\omega_x u_k - \omega_y v_k$, as a function of the motion parameters. In [99], the peaks of the periodograms of $F(\omega_x, 0; t)$, $F(0, \omega_y; t)$ and $F(\omega_x, \omega_y; t)$ are determined to estimate the motion parameter sets in the x and y directions, and the set consisting of the sum of these two velocity components, respectively. Then the motion parameter sets in the x and y directions are paired such that,

$$\sum_{i=1}^M (\tilde{u}_i + \tilde{v}_i - (\widetilde{u+v})_i)^2 \quad (2.40)$$

is minimized where \tilde{u}_i , \tilde{v}_i and $(\widetilde{u+v})_i$ are members of the sets consisting of the motion parameters in the x and y directions and their sums, respectively [99]. The parameter M indicates the number of objects in the scene with distinct motion parameters and is estimated as the maximum of the number of elements of the three velocity sets described above.

This approach may give erroneous results in some cases. For example, if we have three objects in our scene moving with motion parameters $(u_1, v_1) = (0, -2)$, $(u_2, v_2) = (-2, 0)$ and $(u_3, v_3) = (-2, -2)$, the above pairing method of [99] will find the number of moving objects in the scene as $M = 2$, since all the velocity sets will contain 2 elements. To be more specific, the set consisting of the x-components of the velocities will be $\{0, -2\}$, the set consisting of the y-components of the velocities will be $\{-2, 0\}$, and the set consisting of the sum $u_l + v_l$ will be $\{-2, -4\}$. In the next section, we propose a novel method based on fuzzy c-means clustering that eliminates the need for pairing the velocity components using (2.40).

2.2.2. Motion Parameter Estimation Using Fuzzy C-Planes Clustering

In this section, we present a new approach for simultaneous motion parameter estimation and pairing which is based on the fuzzy c-planes clustering algorithm [101, 102, 83]. The presented method is based on fitting planes to the 3-D data in the frequency domain using fuzzy c-planes algorithm. The parameters of these planes

correspond to the horizontal and vertical motion parameters of the objects in the scene. This approach overcomes the difficulties of the pairing method of [99].

For different $(\omega_{x_i}, \omega_{y_i})$ pairs, the peaks of the periodogram of (2.39) are estimated and the 3-D data are formed as:

$$\mathbf{h}_i = [\omega_{x_i}, \omega_{y_i}, \omega_i]^T, \quad (2.41)$$

where ω_i denotes the location of a peak. There may be more than one dominant peak for a specific $(\omega_{x_i}, \omega_{y_i})$ pair.

Note that since these 3-D points satisfy the equation

$$\omega_i = -\omega_{x_i} u_k - \omega_{y_i} v_k, \quad k \in \{1, \dots, M\} \quad (2.42)$$

they are expected to form M planes in three dimensional frequency domain. The plane parameters are the negatives of the motion parameters. We estimate the plane parameters via clustering these 3-D data into M planes. The estimated plane parameters give us the motion parameters without the need for a pairing step.

In order to cluster the above 3-D frequency domain points into M planes, we use the fuzzy c-planes algorithm (FCP) [101, 102, 83]. The FCP algorithm uses all of the sampled 3D data vectors as given in (2.41), for different values of (ω_x, ω_y) , instead of some specific values of (ω_x, ω_y) . The FCP is a probabilistic clustering method which minimizes the following cost function:

$$J_{FCP} = \sum_{i=1}^K \sum_{j=1}^M m_{ij}^q D_{ij}^2, \quad (2.43)$$

where K is the total number of 3D data points, M is the number of planes, m_{ij} is the membership value of the i^{th} data vector to the j^{th} plane, D_{ij} is the distance between the i^{th} data vector and the j^{th} plane, and q is the fuzzifier which is a weight parameter.

The FCP method is “fuzzy” in the sense that at each iteration, a data vector belongs to each of the classes to a certain degree. The above cost function denotes the weighted average of the distance between the data points and their assigned planes. The distance is calculated as given in (2.48), where \mathbf{c}_j is a vector which characterizes normalized centroid of the j^{th} plane. The membership parameter is subject to the constraints given in (2.44) and (2.45). In order to minimize the cost function (2.43), the derivatives of J_{FCP} with respect to D_{ij} and \mathbf{c}_j are calculated, which give us the expressions (2.46) and (2.49) (for details see [83]). The steps of the alternating optimization algorithm that minimizes the cost function (2.43) are given below:

- I. Initialize each membership value m_{ij} , randomly such that the constraints,

$$0 \leq m_{ij} \leq 1, \quad i = 1, \dots, K, \quad j = 1, \dots, M, \quad (2.44)$$

and

$$\sum_{j=1}^M m_{ij} = 1, \quad \forall i, \quad (2.45)$$

are satisfied. Inspecting the 3D plot of the data vectors (\mathbf{h}_i 's) visually to set simple thresholds may also be helpful in initializing the algorithm close to the desired local minimum when the data are noisy.

- II. Estimate each normalized cluster centroid, \mathbf{c}_j using the equation,

$$\mathbf{c}_j = \frac{\sum_{i=1}^K m_{ij}^q \mathbf{h}_i}{\sum_{i=1}^K m_{ij}^q}, \quad j = 1, \dots, M. \quad (2.46)$$

Here, \mathbf{h}_i denotes the data vectors as defined in (2.41), and q is a parameter that represents the amount of fuzziness of the clustering. The larger the q is, the fuzzier the clustering becomes, since the weighting factor m_{ij}^q will become closer to zero faster. For example, if $m = 0.9$ and $q = 10$, then the weight m^q will be 0.3487. If q is chosen to be large, this may cause the (local) minima of the cost function to be less clearly distinguishable. However, choosing a large value for

q may help to reduce the effect of noisy data by minimizing its weight [101]. A frequently chosen value for q is 2 [83].

III. Estimate the covariance matrix of each cluster (plane), \mathbf{R}_j , as follows:

$$\mathbf{R}_j = \frac{\sum_{i=1}^K m_{ij}^q (\mathbf{h}_i - \mathbf{c}_j)(\mathbf{h}_i - \mathbf{c}_j)^T}{\sum_{i=1}^K m_{ij}^q}, \quad j = 1, \dots, M. \quad (2.47)$$

IV. Find the distance of each data point to each of the current clusters (planes). These distances will be used in the next step to update the membership values. Since the data points of each cluster are expected to lie on a plane, which is spanned by two vectors, the largest two eigenvectors of \mathbf{R}_j (\mathbf{e}_{j1} and \mathbf{e}_{j2}) are found and the squared distance between the data point i and the plane j is estimated as follows:

$$D_{ij}^2 = \|\mathbf{h}_i - \mathbf{c}_j\|^2 - \sum_{k=1}^2 ((\mathbf{h}_i - \mathbf{c}_j)^T \mathbf{e}_{jk})^2, \quad \forall i, j. \quad (2.48)$$

The above equation calculates the perpendicular distance from the data point \mathbf{h}_i to the estimated plane j . The first term is the square of the distance between the data point h_i and the cluster centroid \mathbf{c}_j . The second term finds the projection of the vector from \mathbf{c}_j to \mathbf{h}_i onto each of the eigenvectors of the plane and subtracts them from the first term to find the perpendicular distance.

V. Update the membership values of the data points as follows:

$$m_{ij} = \frac{(D_{ij}^2)^{\frac{-1}{q-1}}}{\sum_{k=1}^M (D_{ik}^2)^{\frac{-1}{q-1}}}, \quad i = 1, \dots, K; \quad j = 1, \dots, M. \quad (2.49)$$

VI. Repeat the steps 2-5 until the maximum change in the membership values or the cluster centroids between two successive iterations is below a certain error threshold ε , i.e.,

$$\max(\|m_{ij}^s - m_{ij}^{s+1}\|) < \varepsilon, \quad (2.50)$$

where s denotes the iteration number.

VII. Estimate motion parameters of each object using the eigenvectors of each plane.

When the number of planes is not known a priori, the numerical value of the cost function J_{FCP} can be used to estimate it. The cost function J_{FCP} decreases for increasing values of M . However, the decrease occurs in larger steps until the correct number of planes is reached, and the cost function decreases in smaller steps afterwards. This property is used to estimate the number of planes as described in the next section.

2.2.3. Experimental Results

The proposed method is first tested using a synthetically generated sequence, which contains three moving blocks on a stationary background. Two frames of this sequence are shown in Figure 2.8. The motion vectors for the three blocks are $(u_1, v_1) = (-2, -2)$, $(u_2, v_2) = (-2, 0)$, $(u_3, v_3) = (0, -2)$ in the northwest direction. The theoretical motion planes in the 3-D frequency domain are shown in Figure 2.9. The top plane corresponds to the motion vector $(u_1, v_1) = (-2, -2)$, and the other planes correspond to the motion vectors $(u_2, v_2) = (-2, 0)$, and $(u_3, v_3) = (0, -2)$. In Figure 2.10, several periodogram plots are given for different (ω_x, ω_y) pairs. Note that three peaks are clearly visible in Figure 2.10(a), whereas only two peaks exist when $\omega_x = \omega_y$ as shown in Figure 2.10(b) and when ω_x or ω_y is close to zero as shown in Figure 2.10(c) and (d). The peaks of the periodogram are estimated using the *pickpeak* function of MATLAB which simply uses the derivatives of the periodogram. Some spurious peaks are eliminated afterwards if the magnitude of the peak is smaller than a certain fraction (0.3) of the maximum peak. The 3-D data extracted from the peak locations of the periodograms are shown in Figure 2.11 which are clustered into three planes. In order to estimate the number of objects, we plotted the cost function (2.43) for different number of planes as shown in Figure 2.13. The cost function decreases sharply until $M = 3$, and the remaining part of the plot is almost flat. This shows that the number of objects in the scene is 3. The estimated velocities are close to the true values as tabulated in Table 2.1 and as demonstrated in Figure 2.12. Using fuzzy c-planes clustering, the velocities of the three moving objects are correctly estimated whereas, this is not possible with the pairing approach of [99] as discussed in Section

2.2.1.

Another sequence that we used for testing the plane-fitting approach is the “Taxi in Garden” sequence, which is shown in Figure 2.14. In this sequence, the white taxi moves with a velocity vector of $(3, -1)$ in the north-east direction. As a preprocessing step, we applied histogram equalization in order to enhance the contrast. Although the background is cluttered, the plane fitting approach was able to estimate the velocity vector as $(2.9948, -0.98736)$.

We also used two natural sequences to test the algorithm. The first one is the “Hamburg Taxi” sequence which is shown in Figure 2.15. The spectrum analyses are done using the MUSIC algorithm [104]. The true velocity values are determined by manual feature point tracking [99]. As seen in Table 2.2, the estimated motion parameters are quite close to the actual values. The first three rows of Table 2.2 gives the motion parameters of the three vehicles and the last row gives the parameters of the stationary background. The walking person in the upper left corner is not detected since it is too small. The cost function (2.43) for different number of planes is shown in Figure 2.13. The cost function decreases sharply until M increases up to 4, and the rest of the plot is almost flat indicating that the number of objects in the scene is 4.

The second natural sequence used for testing the algorithm is the “Coast Guard” sequence, which is shown in Figure 2.16. Between the 130th and the 140th frames, the boat moves towards right and the background translates towards left due to the motion of the camera. The true values of the motion vectors are estimated by manual feature point tracking. Table 2.3 shows the true and estimated values of the motion vectors. The first row of Table 2.3 gives the motion parameters of the boat and the second row gives the parameters of the background. Note that although the motion in the scene is not purely translational due to the waves in front of the boat, the flag and the person on the boat, the estimated motion vectors are reasonably close to the true values.

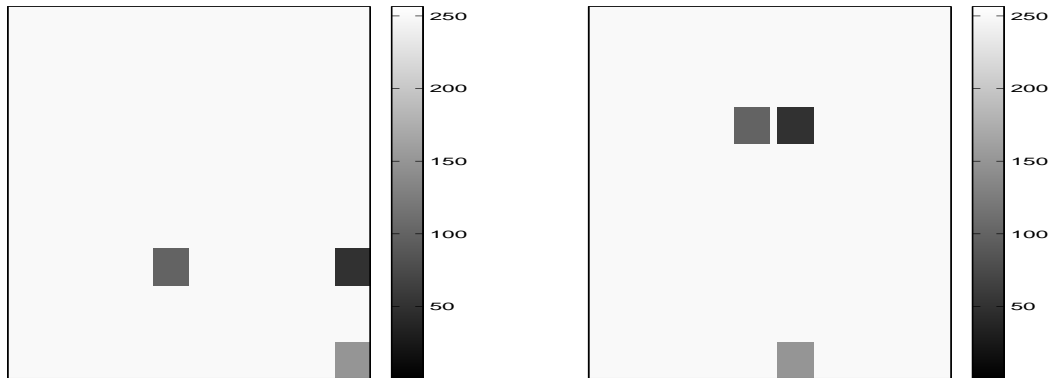


Figure 2.8. The 1st and the 20th frames of the sequence containing three moving blocks

Table 2.1. The original and estimated motion vectors for the sequence containing three moving blocks

ORIGINAL VECTORS	ESTIMATED VECTORS
$(u_1, v_1) = (-2, -2)$	$(\hat{u}_1, \hat{v}_1) = (-2.018, -2.018)$
$(u_2, v_2) = (-2, 0)$	$(\hat{u}_2, \hat{v}_2) = (-2.060, -0.071)$
$(u_3, v_3) = (0, -2)$	$(\hat{u}_3, \hat{v}_3) = (0.009, -2.044)$

Table 2.2. The original and estimated motion vectors for the “Hamburg Taxi” sequence

ORIGINAL VECTORS	ESTIMATED VECTORS
$(u_1, v_1) = (2.7, 0.45)$	$(\hat{u}_1, \hat{v}_1) = (2.971, 0.618)$
$(u_2, v_2) = (-2.65, -0.35)$	$(\hat{u}_2, \hat{v}_2) = (-2.243, -0.334)$
$(u_3, v_3) = (-0.83, -0.45)$	$(\hat{u}_3, \hat{v}_3) = (-1.0782, -0.723)$
$(u_4, v_4) = (0, 0)$	$(\hat{u}_4, \hat{v}_4) = (0.0015, -0.005,)$

Table 2.3. The original and estimated motion vectors for the “Coast Guard” sequence

ORIGINAL VECTORS	ESTIMATED VECTORS
$(u_1, v_1) = (0.36, -0.18)$	$(\hat{u}_1, \hat{v}_1) = (0.3014, -0.1334)$
$(u_2, v_2) = (-1.52, -0.2)$	$(\hat{u}_2, \hat{v}_2) = (-1.7005, -0.3907)$

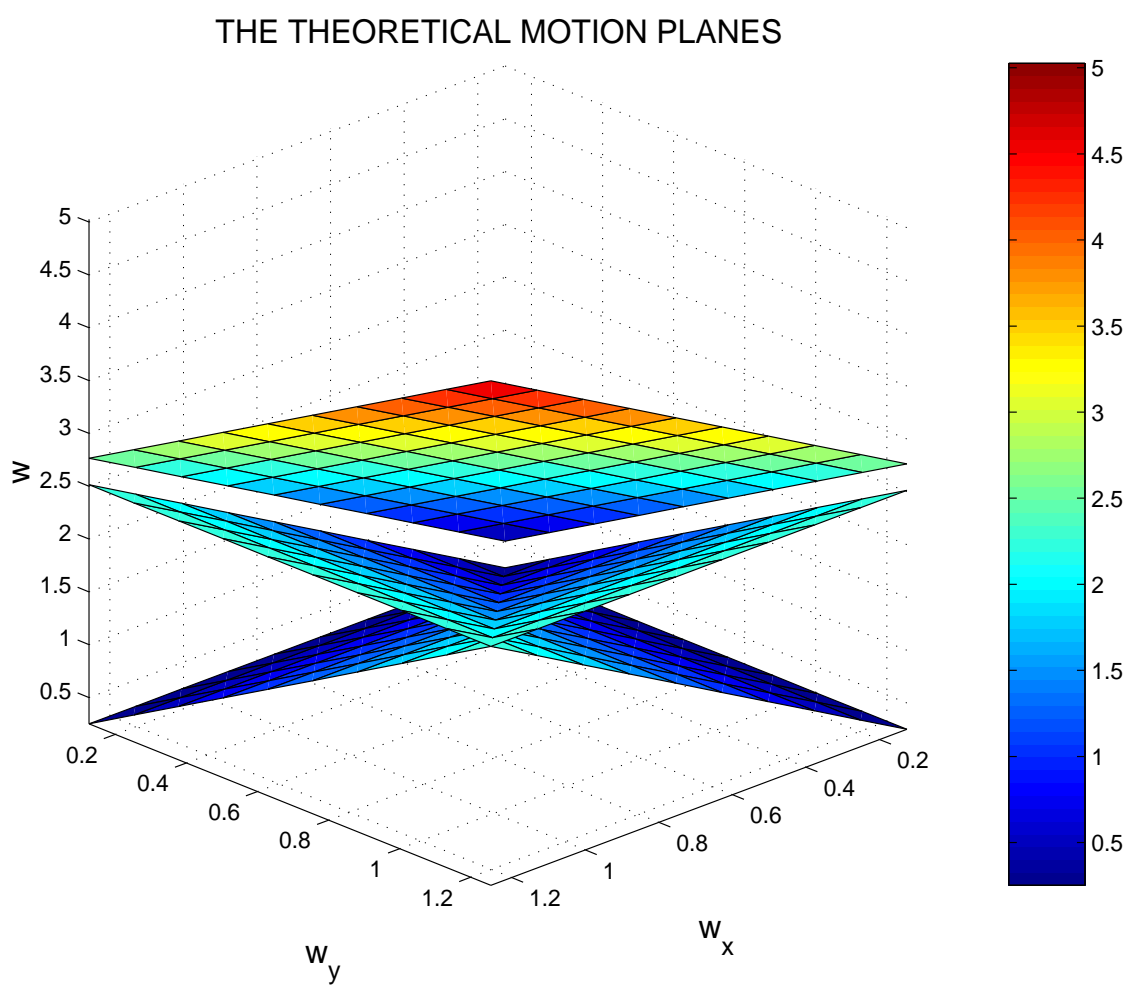


Figure 2.9. The theoretical motion planes for the sequence containing three moving blocks

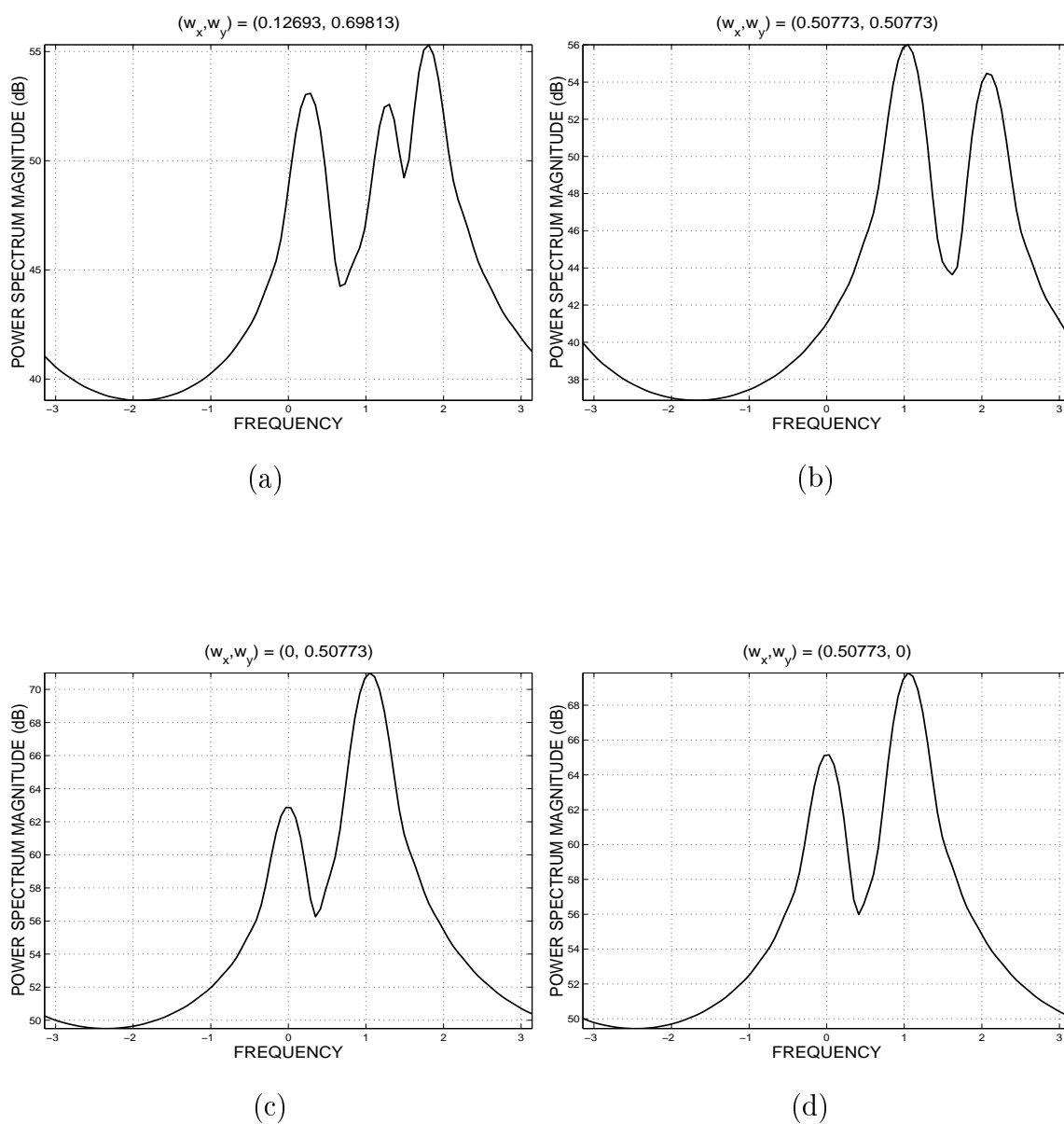


Figure 2.10. The periodogram plots for (a) $(\omega_x, \omega_y) = (0.127, 0.698)$, (b) $(\omega_x, \omega_y) = (0.508, 0.508)$, (c) $(\omega_x, \omega_y) = (0, 0.508)$, (d) $(\omega_x, \omega_y) = (0.508, 0)$

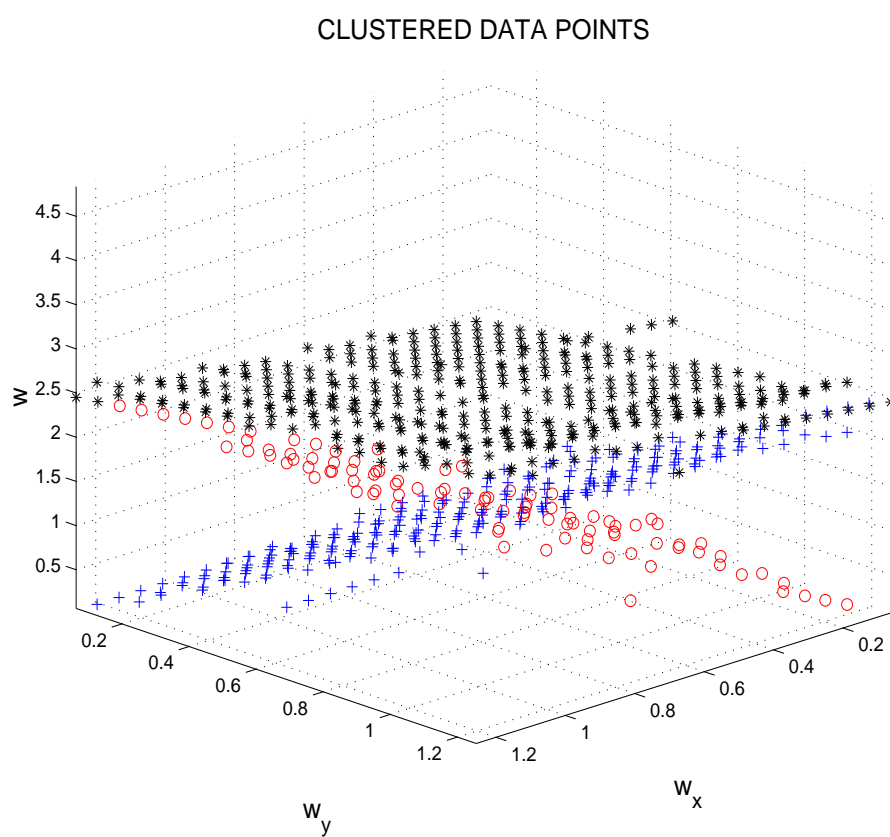


Figure 2.11. The data points clustered into planes using the FCP algorithm

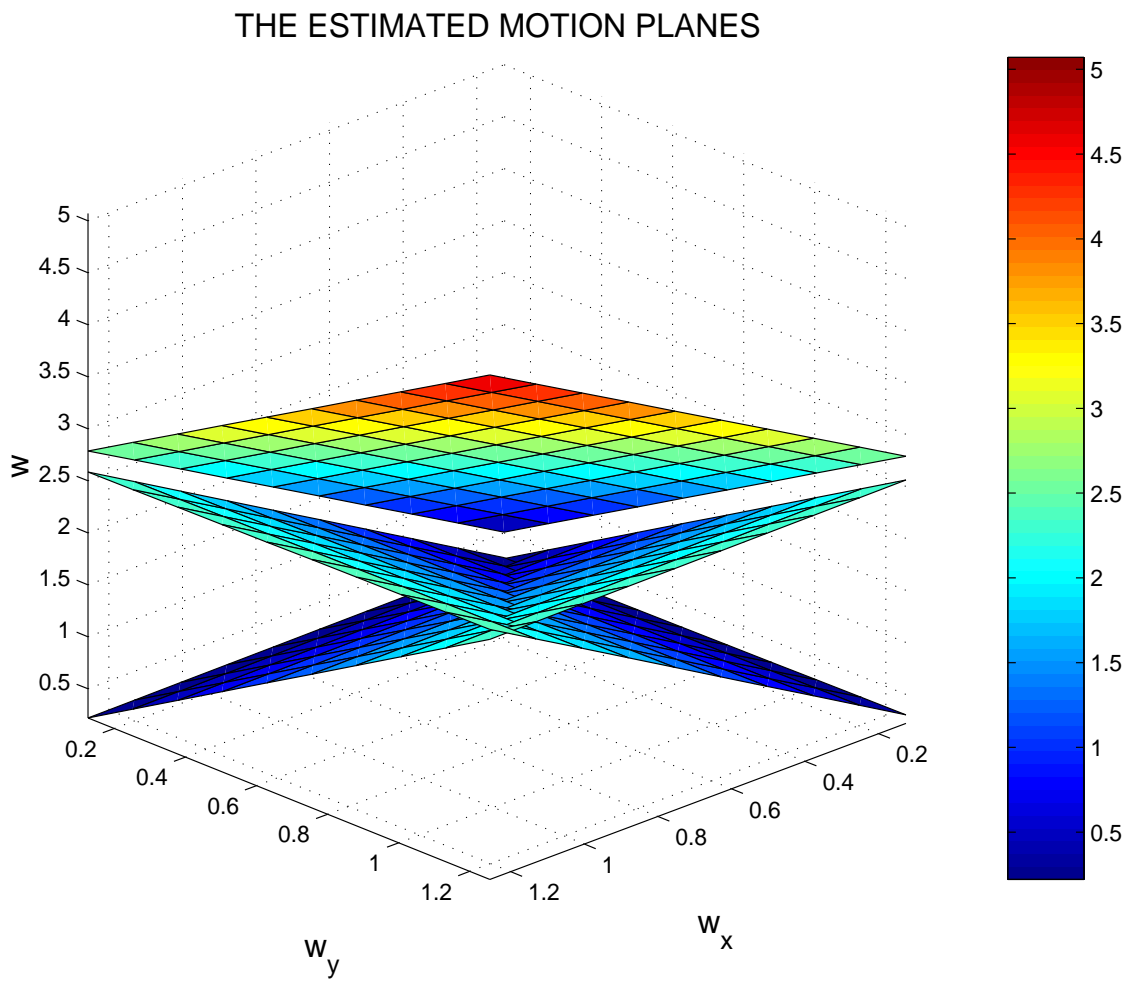


Figure 2.12. The motion planes estimated using the clustered data shown in Figure 2.11

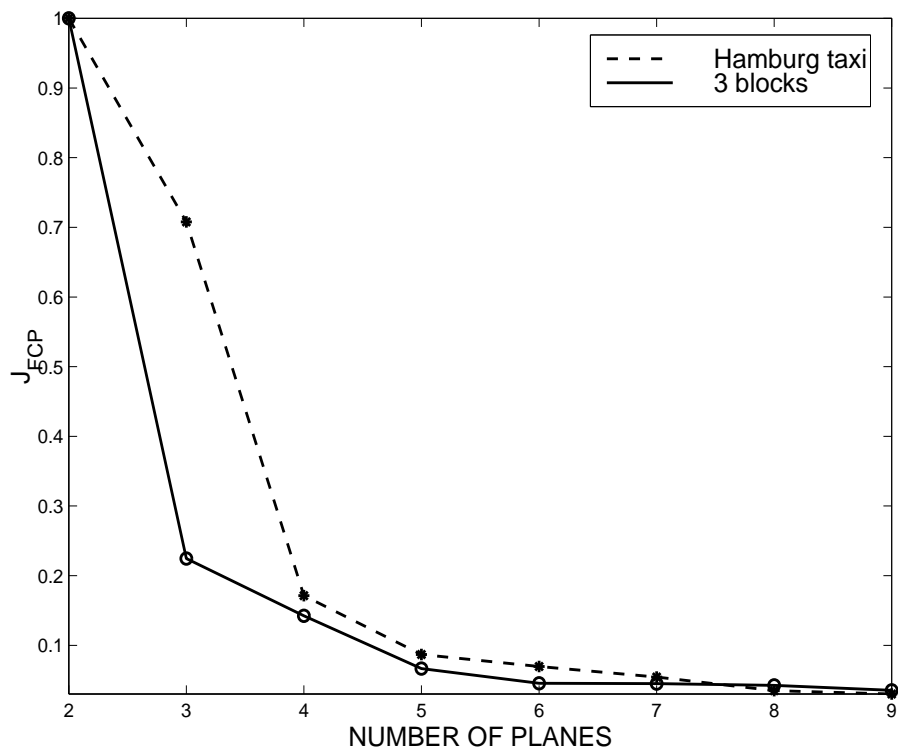


Figure 2.13. The FCP cost function versus number of planes. The magnitudes have been scaled so that the maximum cost value is mapped to 1

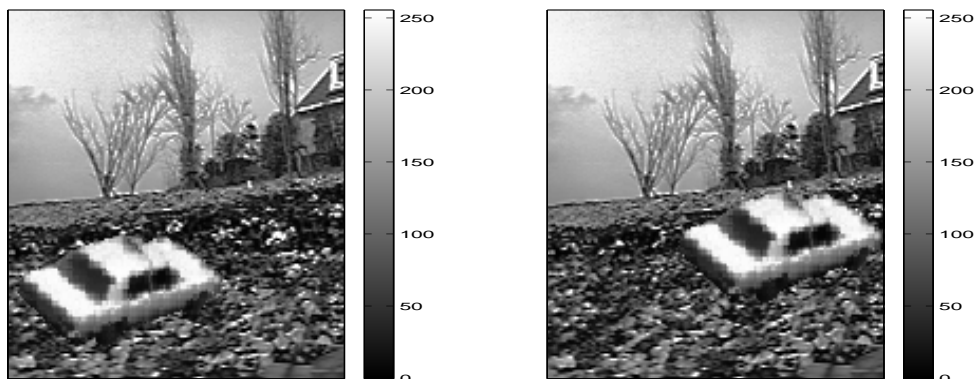


Figure 2.14. The 1st and the 20th frames of the “Taxi in garden” sequence



Figure 2.15. The 1st and the 20th frames of the “Hamburg Taxi” sequence



Figure 2.16. The 130th and the 140th frames of the “Coast Guard” sequence

3. PERFORMANCE EVALUATION MEASURES FOR OBJECT-BASED VIDEO SEGMENTATION

Many object-based segmentation algorithms have been proposed in the literature recently [27, 29, 28, 20]. These algorithms use different sets of techniques and result in different performance. Furthermore, their performances vary with different image sequences depending on the content of the sequences. Their comparative assessment is often based upon subjective judgement of a few frames in known sequences. In this context, an objective spatio-temporal segmentation measure would be a very useful basis of comparison of algorithmic performance. Furthermore, the time and motion consistency of object-based video segmentation algorithms must also be put into evidence. Although there are many approaches, e.g., [81, 75, 76, 77, 78, 79, 80], that try to evaluate image segmentation quantitatively, they are all for still images, and not for video sequences. However, recently, several performance measures using ground-truth data has been used by [105] which utilize the trajectories of the centroids of the tracked objects and their velocities.

In this chapter, we propose performance measures to quantitatively evaluate the empirical results of spatio-temporal video segmentation methods. The measures address the cases both when ground-truth video object planes are available and when they are unavailable. The proposed evaluation measures using ground-truth data are introduced in Section 3.1. The performance evaluation measures without ground-truth data are presented in Section 3.2. In Section 3.3 we try to observe the behaviour of the measures with and without ground-truth segmentation masks over the same data set. In Section 3.4, the statistical behaviour of these performance measures are analyzed through principal component analysis and canonical correlation analysis.

3.1. Performance Evaluation Measures Based on Ground-Truth Data

For still image segmentation, many algorithms have been proposed to evaluate their performance in the absence of ground-truth data [77, 79, 78, 75, 81] and when

the ground-truth is available [76, 79, 81]. The ground-truth for a still image defines the support of the object(s), which corresponds to the location, size, and the shape of the object (but not the inner texture). For video segmentation, the ground-truth must be defined differently for intended applications. If we directly extend the definition of ground-truth for still images to video, then the ground-truth for an image sequence consists of the support of the object(s) in each frame of the sequence. Most MPEG-4 applications require the motion information of the objects, which can be in the form of dense motion field, motion along the object boundaries or affine motion parameters for each object. Therefore, in the final analysis, the comparison measures should reflect the fidelity of the spatio-temporal evolution of the segmented objects. In the following we introduce four quantitative measures for evaluation of object-based segmentation [106]. We use measures for shape, pixel and motion consistency separately, which is similar to the coding strategy of MPEG-4, since each object's shape is coded, then its texture is coded and motion is coded separately. So, our measures match to the MPEG-4 requirements.

We assume that the frames contain N pixels and M objects. When the ground truth (spatial support and motion information of the objects) is available, the following distance measures between the ground truth and the segmented video object planes can be considered:

3.1.1. Misclassification Penalty (DP)

The misclassified pixel measure used for still image segmentation is adapted to compare the estimated segmentation map of each frame to its ground-truth version on an object-by-object basis. Misclassified pixels between ground-truth objects and test objects can be penalized as a function of their distance from the ground-truth boundary as discussed in [80] and [107]. Another approach presented here, uses the Chamfer distance transformation as follows:

Let the object set of the ground-truth (reference) image and of the segmented image be denoted as $G = \{g_i, i = 1, \dots, M\}$ and $s = \{s_i, i = 1, \dots, M\}$. Let the

label functions $L_G(x, y; t) \in \{1, \dots, M\}$ and $L_S(x, y; t) \in \{1, \dots, M\}$ denote the objects to which the pixel at location (x, y) belongs to at time t , in the ground-truth and segmented frames, respectively. We define the indicator function $I_i(x, y; t)$ at pixel location (x, y) at frame t as:

$$I_i(x, y; t) = \begin{cases} 0, & L_G(x, y; t) = L_S(x, y; t) = i \\ 1, & \text{else.} \end{cases} \quad (3.1)$$

The discrepancy between two objects g_i and s_i due to misclassified pixels can be calculated as:

$$0 \leq DP_i(t) = \frac{\sum_{(x,y)} I_i(x, y; t) \text{Cham}_{g_i}(x, y; t)}{\sum_{(x,y)} \text{Cham}_{g_i}(x, y; t)} < 1 \quad (3.2)$$

where $\text{Cham}_{g_i}(x, y; t)$ is the modified Chamfer distance of the pixel from the boundary of g_i . The modified Chamfer distance can be found using Chamfer 3-4 mask [27]. Chamfer 3-4 distance transform is an approximation of the Euclidean distance that is implemented using a top-down and a bottom-up pass with the template matrix:

$$\begin{bmatrix} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{bmatrix}. \quad (3.3)$$

First, the binary ground-truth boundary image is processed such that the object boundary pixels are denoted with 0 values and other pixels are assigned a very large numerical value. Then, the above matrix is centered on each pixel of the boundary image, and the sum of the matrix elements and the corresponding image pixels are found. Finally, the minimum of these summations is assigned to the center pixel. We change the labels of the pixels on the boundary as 1 after applying the Chamfer 3-4 transform to obtain the modified Chamfer 3-4 transform.

An example of a modified Chamfer 3-4 transform is given below, where the 0

pixels on the left matrix denotes edge positions:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{\text{Chamfer}} \begin{bmatrix} 3 & 1 & 3 & 4 & 7 \\ 6 & 3 & 1 & 3 & 6 \\ 6 & 3 & 1 & 3 & 6 \end{bmatrix}. \quad (3.4)$$

The segmentation error averaged over all objects becomes:

$$DP(t) = \frac{1}{M} \sum_{i=1, \dots, M} DP_i(t). \quad (3.5)$$

Several other variations can be considered such as weighting the object errors proportionally to their size or importance with respect to other criteria before averaging. Finally, the spatio-temporal distortion reflecting the misclassification penalty over T frames can be calculated as:

$$\overline{DP} = \tau(DP(t)), \quad (3.6)$$

where $\tau(DP(t))$ is some error function, for example,

$$\begin{aligned} \tau(DP(t)) &= \frac{1}{T} \sum_{t=1}^T DP(t)^2 \\ \tau(DP(t)) &= \frac{1}{T} \sum_{t=1}^T DP(t) \\ \tau(DP(t)) &= \max\{DP(t)\}. \end{aligned}$$

3.1.2. Shape Penalty (DS)

A discriminative shape measure is the turning angle function (TAF) of the object boundaries [108]. Turning angle is a function that measures the angle of the counter clockwise tangent with respect to the x-axis as a function of the arc-length. As we walk on the boundary, the TAF increases if we turn left, and the TAF decreases if turn right. We extend the calculation of the TAF from polygons [108] to any arbitrarily shaped

object by fitting a continuous curve to the boundary pixels and taking samples at equal intervals on this curve. Note that this shape measure TAF, is scale and translation invariant, although we do not require it to be so, since we want our object to be in the same position and scale as our ground-truth object. An example of a turning angle function in degrees is shown for a square in Figure 3.1. Another example for a non-polygon shape is given in Figure 3.2, where the starting point is shown with a red square. Note that for closed boundaries the total turning angle (the difference between the first and the last point of the TAF) is always 360 degrees. The distance between

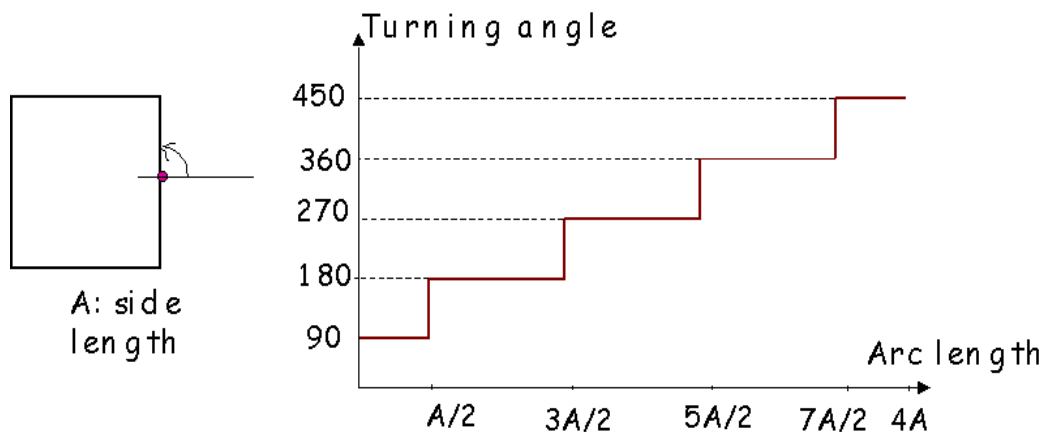


Figure 3.1. The turning angle function of a square

the M object shapes at time t can be calculated from:

$$0 \leq DS(t) = \frac{1}{M} \sum_{i=1}^M DS_i(t) \leq 1 \quad (3.7)$$

$$DS_i(t) = \frac{\sum_{j=1}^K |\Theta_{g_i}^t(j) - \Theta_{s_i}^t(j)|}{2\pi K}, \quad (3.8)$$

where $\Theta_{g_i}^t(\cdot)$ and $\Theta_{s_i}^t(\cdot)$ denote the turning angle functions (TAF) of the i^{th} ground-truth object boundary (g_i) at time t , and of the corresponding estimated video object boundary (s_i) respectively. K denotes the length of the turning angle function, which is equal to the number of boundary pixels. For independence of the starting point during TAF calculation, $\Theta_{g_i}^t(j)$ or $\Theta_{s_i}^t(j)$ can be circularly shifted around j . The difference is re-calculated for each shift to detect the value which gives the minimum TAF difference. For rotation independence, one of the TAF's can be shifted up and down by a certain angle. The angle which gives the minimum TAF difference is the

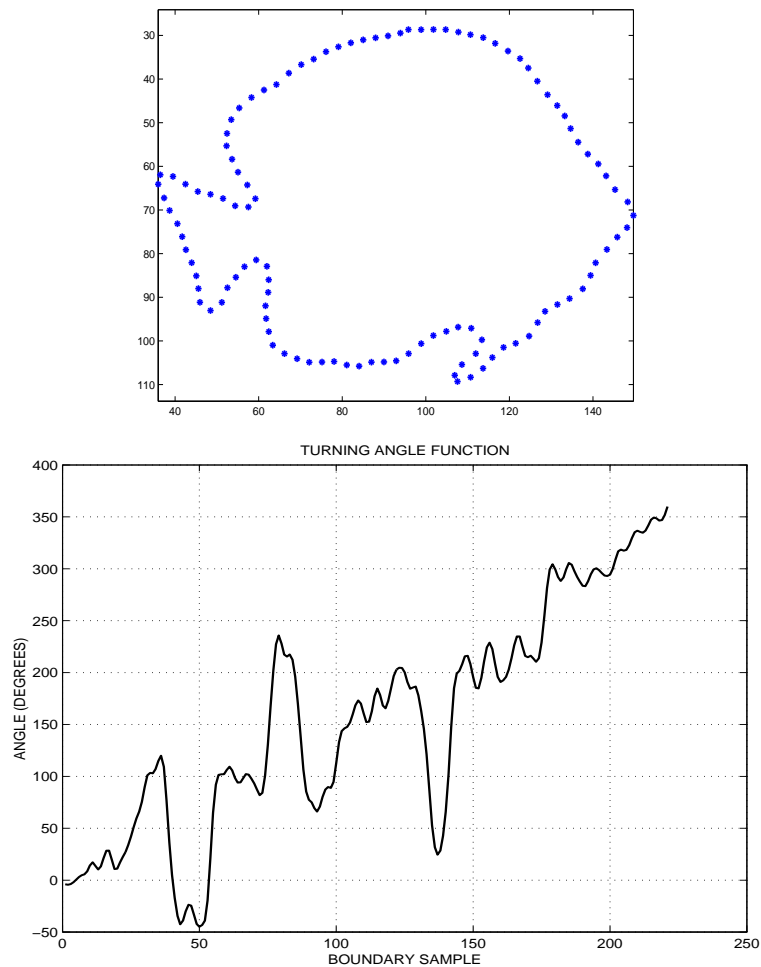


Figure 3.2. The turning angle function of a fish boundary

rotation angle between the two shapes. However, since we know that the rotation angle between the ground truth object and the segmented object should be zero, we generally do not need this step.

The discrepancy between two objects can be weighted by object area before averaging. The shape distortion of a sequence can be computed as in the misclassification penalty, that is, some weighted averaging over the interval T :

$$\overline{DS} = \frac{1}{T} \sum_{t=1}^T \tau(DS(t)) \quad (3.9)$$

3.1.3. Motion Penalty (DM)

Let $\mathbf{v}_{g_i}(t)$ and $\mathbf{v}_{s_i}(t)$ denote the motion information at frame t of the ground-truth object g_i and segmented object s_i , respectively. \mathbf{v} itself can be parametric, such as the affine motion parameters of the object, or it may represent the motion field vectors at each pixel or a neighborhood of pixels. For video objects, it is more significant to compare the consistency of motion trajectories over time. Thus, one can have:

$$DM_i = \tau(\mathbf{v}_{g_i}(t), \mathbf{v}_{s_i}(t)) \quad (3.10)$$

where $\tau(.,.)$ is some distance function between two time evolution surfaces described by \mathbf{v}_{g_i} and \mathbf{v}_{s_i} . For example, one can consider:

$$DM_i = \frac{1}{T} \sum_{t=1}^T DM_i(t) \quad (3.11)$$

$$0 \leq DM_i(t) = \frac{\|\mathbf{v}_{g_i}(t) - \mathbf{v}_{s_i}(t)\|}{\|\mathbf{v}_{g_i}(t)\| + \|\mathbf{v}_{s_i}(t)\|} \leq 1. \quad (3.12)$$

When we want to compare the dense motion fields, it is reasonable to compare the motion on the common area of the ground truth and the segmented objects to eliminate the effects of the shape distortions. Alternatively, we can estimate affine motion parameters using the dense motion field and the support of the objects and compare them.

Finally, the object trajectory scores can be averaged over all objects:

$$\overline{DM} = \frac{1}{M} \sum_{i=1}^M \tau_2(DM_i). \quad (3.13)$$

3.1.4. Combined Penalty (CP)

A weighted averaging of more than one criterion can be used to attain a more comprehensive quality measure of the spatio-temporal segmentation with respect to its

ground-truth:

$$CP = \gamma_1 \overline{DP} + \gamma_2 \overline{DS} + \gamma_3 \overline{DM}, \quad (3.14)$$

where the parameters γ_1, γ_2 and γ_3 can be adjusted to give different weights to misclassification penalty, shape penalty and motion penalty, respectively. For example, if we know that our object is rigid such as a taxi, we know that its shape will not change much. Therefore, we can give a high weight to the shape penalty. If the texture of the object is important we can give more weight to the misclassification penalty. However, in selecting the weighting parameters we need to be careful about the numerical ranges of each measure \overline{DP} , \overline{DS} and \overline{DM} . Although they are all between 0 and 1, their numerical dynamic range can differ. In order to eliminate data range effects when comparing the performance of several algorithms, we can normalize the worst score in each measure to be one, as will be done in the next section. The weighting parameter selection can be carried out after this normalization.

3.1.5. Performance Evaluation of the Video Segmentation Using Ground-Truth Segmentation Maps

The proposed performance evaluation measures are used to evaluate three essentially different approaches for video segmentation, which were described in detail in Section 1.3.

- A region-based parametric motion segmentation algorithm [28] (Method 1);
- An algorithm that finds the best match of the binary model of the object(s) in the edge map of subsequent frames [27] (Method 2);
- An interactive algorithm which finds the initial regions using fuzzy c-means clustering of the feature vectors of each pixel [29] (Method 3).

The proposed performance evaluation measures are tested on one synthetically generated sequence and one real sequence, named “Taxi in the garden” and “Hamburg taxi,” respectively. A sample frame from each sequence is shown in Figure 3.3. In

order to generate the synthetic sequence, the white car extracted from the “Hamburg taxi” sequence is mounted on the first frame of the “flower garden” sequence and the car is moved with a given motion vector of $(3, -1)$ in the x and y directions. The segmentation results for the second frame of the “taxi in the garden” sequence using the three algorithms [28, 27, 29] are given in Figure 3.4. The performance measures proposed in Section 3.1 are evaluated and plotted for 15 frames in Figure 3.6. We can observe in Figure 3.6 that Method 3 has the worst $DP(t)$ scores and Method 2 has the best $DS(t)$ scores for almost all the frames, which are in agreement with the subjective comparisons on Figure 3.4.

The results for both sequences are summarized in Table 3.1 where the performances measures of the three object-based video segmentation methods are averaged over all frames. The scores are normalized such that the worst score for a measure among the three methods is assigned to 1. CP is found by giving equal weights to \overline{DP} , \overline{DS} , and \overline{DM} . For the “taxi in the garden” sequence, Altunbaşak-Eren-Tekalp method performs best in terms of the misclassification penalty and the motion penalty. However, its shape penalty score is high. This is due to the busy background of the flower garden sequence. For the same reason, the shape penalty and misclassification penalty scores of Castagno-Ebrahimi-Kunt method are also high. In terms of motion penalty, the Meier-Ngan method performs the worst since it finds the motion of the edge-model of the car by searching in the edge map of the next frame, which is quite crowded because of the flowers.

The segmentation results for the fourth frame of the “Hamburg taxi” sequence are given in Figure 3.5. In the “Hamburg taxi” sequence which contains multiple objects, the relative performance results do not change much. Although the quantitative performances can change with the parameters chosen in the individual algorithms and the test sequences, Altunbaşak-Eren-Tekalp performed better than the other algorithms in our experiments and the other two algorithms closely follow it.



Figure 3.3. (a) The second frame of the “Taxi in the garden” sequence (b) The fourth frame of the “Hamburg taxi” sequence

Table 3.1. The performances of the three object-based video segmentation methods averaged over all frames

Criteria	Taxi in the garden sequence			Hamburg taxi sequence		
	Method 1	Method 2	Method 3	Method 1	Method 2	Method 3
\overline{DP}	0.16	0.33	1	0.72	0.41	1
\overline{DS}	0.69	0.15	1	0.83	0.98	1
\overline{DM}	0.08	1	0.1	0.15	1	0.24
CP	0.31	0.49	0.7	0.57	0.79	0.74

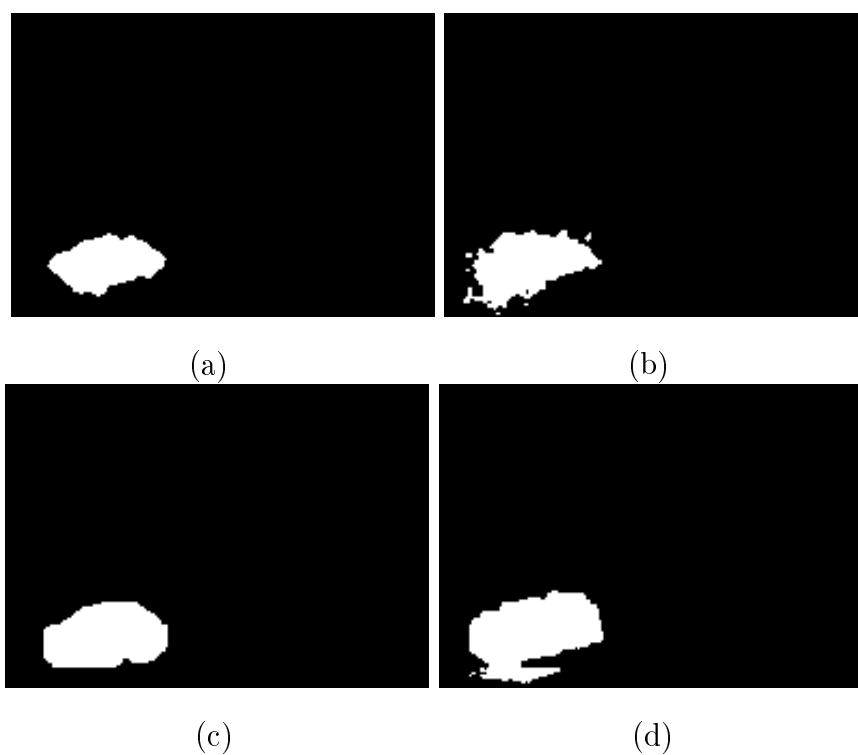


Figure 3.4. (a) The ground truth VOP for frame 2 of the “Taxi in the garden” sequence. (b), (c), (d) The segmentation results of Method 1, 2, and 3, respectively

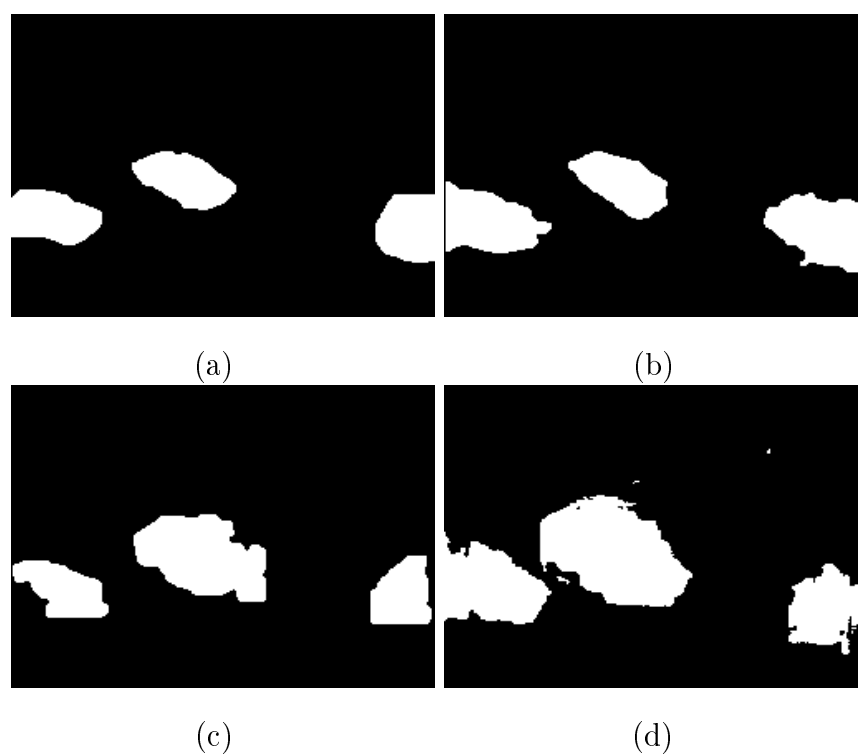
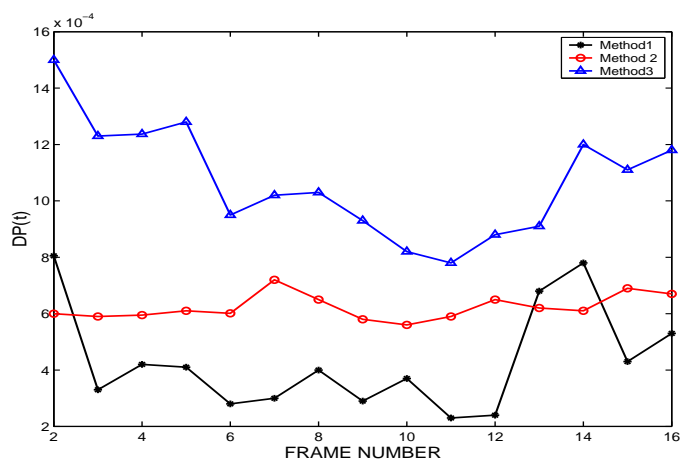
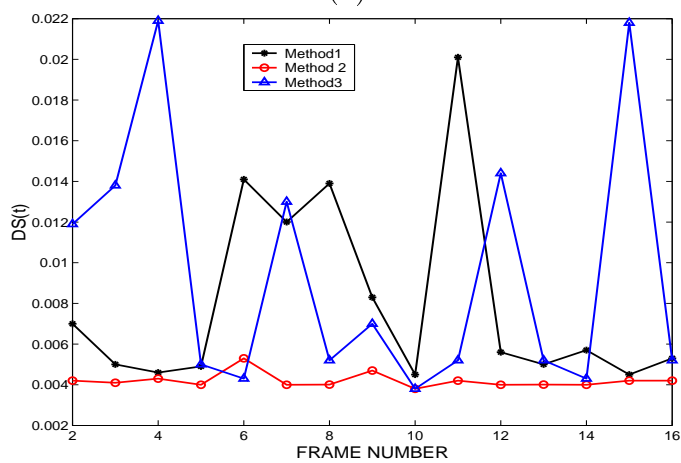


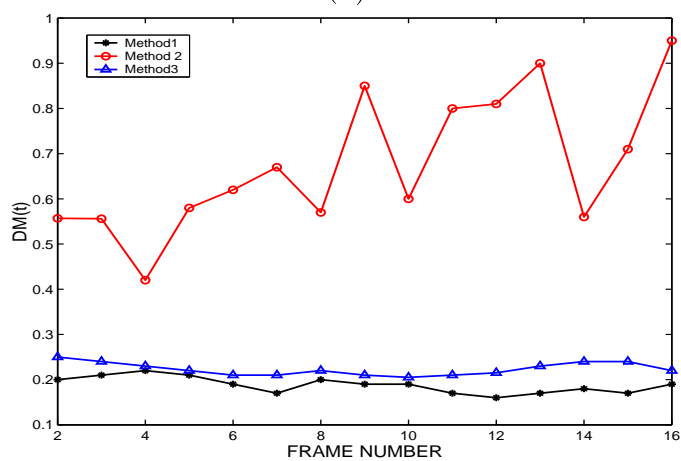
Figure 3.5. (a) The ground truth VOP for the fourth frame of the “Hamburg taxi” sequence (b), (c), (d) The segmentation results of Method 1, 2, and 3, respectively



(a)



(b)



(c)

Figure 3.6. (a) The misclassification penalty $DP(t)$ (b) The shape penalty $DS(t)$ (c) The motion penalty $DM(t)$ for the “Taxi in the garden sequence”

3.2. Performance Evaluation Measures Without Ground-Truth Data

The performance evaluation measures proposed in the previous section are useful if ground-truth segmentation maps are available. However, such ground-truth data is very time-consuming and tedious to extract by hand. In fact, it would be impractical to try to obtain such data for all sequences. Therefore, there is a very pressing need to establish non-ground-truthed measures. The aim of this section is to evaluate the performance of the object tracking and segmentation methods quantitatively, when ground-truth segmentation maps for each frame are not available [106]. To this effect, we present measures based on intra-frame and inter-frame color and motion features of the segmented video object planes. Often a single numerical measure is not sufficient to evaluate a segmentation/tracking result, since certain parts (spatially or temporally) of the object boundary can be better segmented/tracked than others depending on the variation of color and motion features between the object and background regions. Hence, we also enable the proposed measure to localize the good and bad segments of the object boundary .

In Section 3.2.1, the intra-frame and inter-frame color measures used for performance evaluation are presented. In Section 3.2.2, the evaluation measure based on the motion features is presented. In Section 3.2.3, a combined measure consisting of color and motion measures is discussed. In Section 3.2.4, experimental results for a video sequence segmented by hand are given to demonstrate the effectiveness of the proposed measures.

3.2.1. Color Measures

The proposed performance measures using color features are based on the following assumptions which are true for most video sequences and are also assumed by many segmentation algorithms: 1) Object boundaries coincide with color boundaries. 2) The color histogram of the object is stationary from frame to frame. 3) The color histogram of the background is different from the color histogram of the object. Note that the background and its color histogram are not restricted to be stationary

from frame to frame. There are also no restrictions on the shape and rigidity of the segmented/tracked object.

Based on the above assumptions, we present two measures for evaluating the fidelity of the segmented video object plane. The first color measure is based on the intra-frame color differences along the estimated object boundary and is presented in Section 3.2.1.1. The second color measure uses the inter-frame color histogram differences and is described in Section 3.2.1.2.

3.2.1.1. Intra-frame Color Differences Along the Object Boundary. In order to evaluate the performance of the tracking algorithm based on the first assumption above, the color of the pixels ‘just inside’ and ‘just outside’ of the estimated object boundary can be compared. In order to define ‘just inside’ and ‘just outside’, we draw short normal lines of length L to the estimated object boundary at equal intervals towards the inside and outside of the object.

A sample VOP for the 32th frame of the “Hall Monitor” sequence is given in Figure 3.7 (a). The points at the ends of the normal lines drawn to the boundary are marked as illustrated in Figure 3.7(b). The marked points are shown with plus signs. A closer look at one of these normal lines is given in Figure 3.7(c). The two points ‘just inside’ and ‘just outside’ of the boundary are shown with symbols \mathbf{p}_I^i and \mathbf{p}_O^i , respectively. We define the color difference measure calculated along the boundary of the object in frame t as:

$$0 \leq d_{CB}(t) = 1 - \frac{1}{K_t} \sum_{i=1}^{K_t} d_{CB}(t; i) \leq 1, \quad (3.15)$$

$$d_{CB}(t; i) = \frac{\|\text{Col}_O^i(t) - \text{Col}_I^i(t)\|}{\sqrt{3 \times 255^2}} \quad (3.16)$$

where, K_t is the total number of normal lines drawn to the boundary of the object at equal intervals in frame t , $\mathbf{p}_O^i(t)$ is the end point of the normal line that is outside the object boundary and $\text{Col}_O^i(t)$ is the average color calculated in the $N \times N$ neighborhood of the pixel $\mathbf{p}_O^i(t)$ using YCbCr color space. The average inside color $\text{Col}_I^i(t)$ is defined similarly. Instead of the averaging operation, the α -trimmed mean can also be used in (3.16), which will be described shortly. When the segmentation is bad, the measure

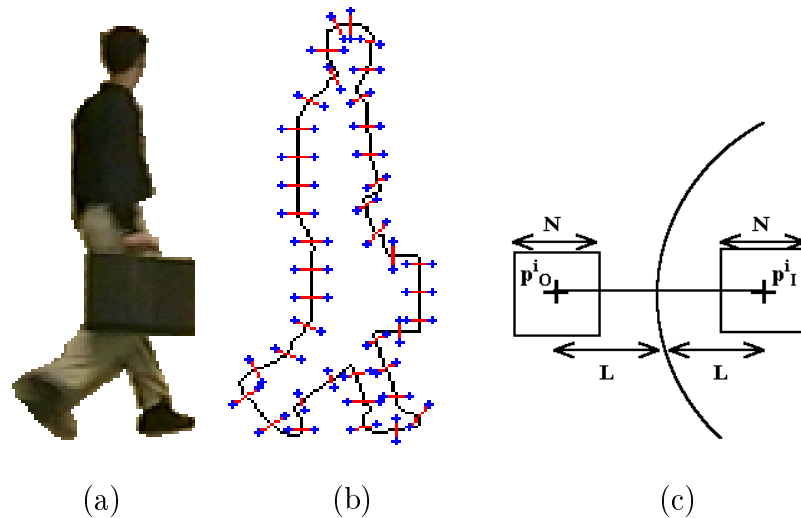


Figure 3.7. (a) A sample VOP (b) The boundary of the VOP with the normal lines
(c) A close-up of a normal line drawn to the boundary

$d_{CB}(t)$ is expected to get high values, which are close to 1.

We define the color measure for the whole sequence as:

$$0 \leq D_{CB} = \tau(d_{CB}(t), t = 1, \dots, T) \leq 1, \quad (3.17)$$

where T is the number of frames in the sequence. The function $\tau(\cdot)$ can be defined in different ways such as the mean function, or the α -trimmed mean function [109]. The α -trimmed mean is defined as:

$$D_{CB,\alpha} = \frac{1}{T - 2[\alpha T]} \sum_{j=[\alpha T]+1}^{T-[\alpha T]} d_{CB}(j), \quad (3.18)$$

where $[\cdot]$ denotes the integer ceiling function and $d_{CB}(j)$ denotes the j^{th} element of the sorted array $d_{CB}(\cdot)$. When α is zero, the above expression is identical to the sample mean. When T is odd and α is 0.5, the α -trimmed mean is the same as the median of the sample set.

When the location of the object boundary is estimated correctly in frame t , we expect the color measure $d_{CB}(t)$ to take a small value (close to zero). However, notice that the converse of this sentence is not necessarily true. That is, if the color

measure $d_{CB}(t)$ has a small value in frame t , this does not necessarily imply that the object boundary is located correctly, especially if the background is highly textured. Therefore, this measure should be used carefully depending on the characteristics of the background surrounding the object.

3.2.1.2. Inter-frame Color Histogram Differencing. In order to test whether the object is segmented/tracked correctly in each frame, we make use of the second assumption stated above, i.e. the color histogram of the object is assumed to be stationary from frame to frame. This assumption also implies that there are not significant inter-frame illumination changes. If a part of the background is included into the segmentation map by mistake, the color histogram of the segmented object is expected to follow the background histogram changes.

We can evaluate the stationarity of the color histogram of the segmented object by calculating the pairwise color histogram difference of the video object planes at time t and $t - 1$ or at time t and the first frame. Another approach to make the histogram differencing more robust to self-occlusions and mild intensity variations is to look at the difference between the color histogram of the video object plane at frame t and the smoothed color histogram of the video object planes for frames $\{t - k, \dots, t - 1\}$. This smoothing can be achieved by simple averaging or median filtering of the corresponding bins in the histograms of object planes in frames $\{t - k, \dots, t - 1\}$.

Let us denote the color histogram of the video object calculated using the YCbCr color space at time t as H_t . The locally smoothed color histogram is calculated using the formula:

$$H_{t,av}(j) = Med\{H_{t-k}(j), \dots, H_{t-1}(j)\}, \quad j = 1, \dots, B, \quad (3.19)$$

where B denotes the total number of bins in the color histogram. The color histogram is represented as a 1-D vector obtained by concatenating the histograms for Y, Cb and Cr components.

The discrepancy between the color histograms H_t and $H_{t,av}$ is estimated using four different distance measures as described below [110, 111], namely the L_1 , L_2 , χ^2 and histogram intersection distance measures. In the following formulae, the scaling parameters r_1 and r_2 are used to normalize the data when the total number of elements in the two histograms are different:

$$r_1 = \sqrt{\frac{N_{H_{t,av}}}{N_{H_t}}}, \quad r_2 = \frac{1}{r_1},$$

$$N_{H_t} = \sum_{j=1}^B H_t(j), \quad N_{H_{t,av}} = \sum_{j=1}^B H_{t,av}(j),$$

$$NS_{H_t} = \sum_{j=1}^B H_t^2(j), \quad NS_{H_{t,av}} = \sum_{j=1}^B H_{t,av}^2(j).$$

I. **The L_1 Distance:** The L_1 distance between the two histograms is calculated and normalized to the range $[0, 1]$ as follows,

$$0 \leq d_{CH_{L_1}}(t) = \frac{\sum_{j=1}^B |r_1 H_t(j) - r_2 H_{t,av}(j)|}{2\sqrt{N_{H_t} N_{H_{t,av}}}} \leq 1. \quad (3.20)$$

II. **The L_2 Distance:** The L_2 distance between the two histograms is calculated and normalized to the range $[0, 1]$ as follows,

$$0 \leq d_{CH_{L_2}}(t) = \sqrt{\frac{\sum_{j=1}^B [r_1 H_t(j) - r_2 H_{t,av}(j)]^2}{NS_{H_t} + NS_{H_{t,av}}}} \leq 1. \quad (3.21)$$

III. **The χ^2 Distance:** is used to compare two binned data sets, and to determine if they are drawn from the same distribution function [111]. It is defined and normalized to the range $[0, 1]$ as follows:

$$0 \leq d_{CH_{\chi^2}}(t) = \frac{\sum_{j=1}^B \frac{[r_1 H_t(j) - r_2 H_{t,av}(j)]^2}{H_t(j) + H_{t,av}(j)}}{N_{H_t} + N_{H_{t,av}}} \leq 1. \quad (3.22)$$

IV. **Histogram Intersection Distance:** To quantify the difference of the two histograms using the histogram intersection method, we define the histogram intersection distance as:

$$0 \leq d_{CH_{HI}}(t) = 1 - HI(H_t, H_{t,av}) \leq 1, \quad (3.23)$$

where, $HI(H_t, H_{t,av})$ determines the number of pixels that share the same color in the two histograms [112]:

$$0 \leq HI(H_t, H_{t,av}) = \frac{\sum_{j=1}^B \min[H_t(j), H_{t,av}(j)]}{\min(N_{H_t}, N_{H_{t-1}})} \leq 1. \quad (3.24)$$

Let $d_{CH}(t)$ denote the histogram difference measure calculated using one of the four distances defined above. We define the histogram difference measure for the whole sequence as:

$$0 \leq D_{CH} = \tau(d_{CH}(t), t = 1, \dots, T) \leq 1, \quad (3.25)$$

where the function $\tau(\cdot)$ can be chosen as discussed in the previous section.

3.2.2. Motion Measure

The assumptions that we make about the motion of the segmented object are as follows: 1) The motion vectors of the object that are ‘just inside’ of the object boundary and the background motion vectors that are ‘just outside’ of the object boundary are different. In other words, motion boundaries coincide with the object boundaries. 2) Background is either stationary or has global motion which shall be compensated for.

In order to quantify how well the estimated object boundaries coincide with actual motion boundaries, we use an approach similar to the one used for color. We draw short normal lines to the boundary at regular intervals as shown in Figure 3.7(b), and we look at the difference of the motion vectors around the points $p_O^i(x, y; t)$ and $p_I^i(x, y; t)$. The motion measure estimated following this approach for frame t can be expressed as

follows:

$$0 \leq d_M(t) = 1 - \frac{\sum_{i=1}^{K_t} d_M(t; i)}{\sum_{i=1}^{K_t} \eta_i} \leq 1, \quad (3.26)$$

$$d_M(t; i) = d(\mathbf{v}_O^i(t), \mathbf{v}_I^i(t)) \cdot \eta_i \quad (3.27)$$

$$0 \leq \eta_i = Rel(\mathbf{v}_O^i(t)) \cdot Rel(\mathbf{v}_I^i(t)) \leq 1, \quad (3.28)$$

where $\mathbf{v}_O^i(t)$ and $\mathbf{v}_I^i(t)$ denote the average motion vectors calculated in a $M \times M$ square around the points $\mathbf{p}_O^i(t)$ and $\mathbf{p}_I^i(t)$, respectively, and $d(\mathbf{v}_O^i(t), \mathbf{v}_I^i(t))$ denotes the distance between the two average motion vectors which is calculated as:

$$0 \leq d(\mathbf{v}_O^i(t), \mathbf{v}_I^i(t)) = 1 - \exp\left(-\frac{\|\mathbf{v}_O^i(t) - \mathbf{v}_I^i(t)\|^2}{\sigma^2}\right), \quad (3.29)$$

where selecting the parameter $\sigma = 1$, causes the distance of two motion vectors to be approximately 0.6, if the magnitude of their difference is 1.

In (3.28), $Rel(\mathbf{v}(t))$ denotes the reliability of the motion vector $\mathbf{v}(t)$ at point $\mathbf{p}(t)$ [40]:

$$Rel(\mathbf{v}(t)) = \exp\left(-\frac{\|\mathbf{v}(t) - \mathbf{v}_b(t+1)\|^2}{2\sigma_m^2}\right) \cdot \exp\left(-\frac{\|\mathbf{c}(\mathbf{p}(t); t) - \mathbf{c}(\mathbf{p}(t) + \mathbf{v}(t); t+1)\|^2}{2\sigma_c^2}\right),$$

where $\mathbf{v}_b(t+1)$ denotes the backward motion vector at location $\mathbf{p}(t) + \mathbf{v}(t)$ in frame $t+1$; $\mathbf{c}(\mathbf{p}(t); t)$ denotes the color at point $\mathbf{p}(t)$ at frame t , and the parameters σ_m, σ_c are chosen freely.

We define the motion measure for the whole sequence as:

$$0 \leq D_M = \tau(d_M(t), t = 1, \dots, T) \leq 1. \quad (3.30)$$

3.2.3. Performance Evaluation of Video Segmentation with Non-Ground-Truth Measures

In this section, we derive a single numerical measure to evaluate the performance of object segmentation and tracking results, as well as spatial and temporal localization of incorrect boundary segments.

3.2.3.1. Combining Color and Motion Measures. A single numerical measure can be obtained to evaluate the performance of spatio-temporal segmentation of a video object by combining the color and motion measures defined above as follows:

$$\overline{D} = \mu_{DB}D_{CB} + \mu_{CH}D_{CH} + \mu_M D_M, \quad (3.31)$$

where the parameters μ_{DB} , μ_{CH} , and μ_M can be adjusted according to the characteristics of the video sequence and the relative importance and accuracy of color and motion features. Note that if the summation $\mu_{DB} + \mu_{CH} + \mu_M$ is restricted to be one, the the measure D takes values between $[0, 1]$. If this measure is above a certain threshold, it is possible to localize incorrect boundary segments in time and space as described next. In selecting the weighting parameters we need to be careful about the numerical ranges of each measure D_{CB} , D_{CH} and D_M . Although they are all between 0 and 1, their numerical dynamic range can differ. In order to eliminate data range effects when comparing the performance of several algorithms, we can normalize the worst score in each measure to be one, as will be done in the next section. The weighting parameter selection can be carried out after this normalization.

Another approach of selecting the coefficients is to make them a function of the measures themselves:

$$D = f_{DB}(D_{CB})D_{CB} + f_{CH}(D_{CH})D_{CH} + f_M(D_M)D_M, \quad (3.32)$$

where the function $f(\cdot)$ can be a soft thresholding function (S-function):

$$f(x) = \begin{cases} 0 & x \leq p_1 \\ \frac{(x-p_1)^2}{(p_2-p_1)(p_3-p_1)} & p_1 \leq x \leq p_2 \\ 1 - \frac{(x-p_3)^2}{(p_3-p_2)(p_3-p_1)} & p_2 \leq x \leq p_3 \\ 1 & x \geq p_3 \end{cases} \quad (3.33)$$

A sample S-function is shown in Figure 3.8. The selection of the parameters is another issue here. A possible choice is to use the mean of the data values as p_2 , and select p_1 and p_3 with the help of the data variance (e.g $p_1 = \text{mean} - 3 \times \text{standard deviation}$).

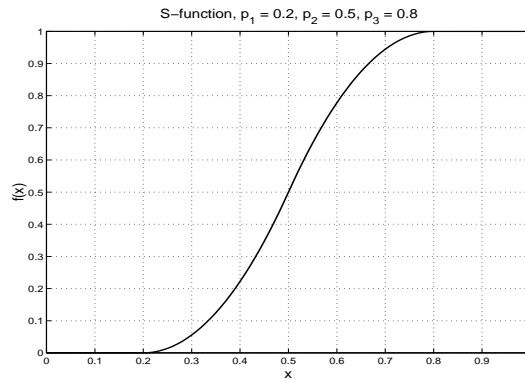


Figure 3.8. A sample S function, $p_1 = 0.2, p_2 = 0.5, p_3 = 0.8$

3.2.3.2. Temporal Localization. The temporal localization can be achieved by checking the color and motion components of the measure

$$d(t) = \mu_{CB,t}d_{CB}(t) + \mu_{CH,t}d_{CH}(t) + \mu_{M,t}d_M(t), \quad (3.34)$$

at each frame against a threshold. A case for temporal localization is illustrated in Figure 3.10.

3.2.3.3. Spatial Localization. In frames t for which $d(t)$ is above the threshold, we can identify the segments of the boundary that have been tracked incorrectly using the color and motion scores that are obtained from ‘inside’ and ‘outside’ point pairs.



Figure 3.9. Frames 32 and 230 of the “Hall Monitor” sequence

Using (3.16) and (3.28), if

$$\gamma_1 d_{CB}(t; i) + \gamma_2 d_M(t; i) < Th, \quad (3.35)$$

where Th is a threshold value, we mark that segment between points $i - 1$ and $i + 1$ of the estimated object boundary as incorrect. An approach of selecting the threshold Th is to calculate the mean and standard deviations of $d_{CB}(t; i)$ and $d_M(t; i)$ values and to set Th as the sum of two (mean - $k \cdot$ standard deviation) values.

3.2.4. Experimental Results with Measures

In order to test the effectiveness of the above proposed measures, we quantitatively evaluate the ground-truth segmentation maps of the Hall monitor sequence which are obtained by hand for frames 32-230. Two sample frames of this sequence are shown in Figure 3.9. A sample video object plane is shown in Figure 3.7(a). In order to observe the sensitivity of the proposed measures to incorrect segmentation maps, we shifted the ground-truth segmentation maps randomly to simulate false segmentation.

3.2.4.1. Experiments with Color Differences Along the Boundary. The color differences along the boundary of the Hall monitor sequence are calculated in the YCbCr color space using the Euclidean distance.

In the first column of Table 3.2, the color measure $D_{CB,\alpha}$ is given which is calculated using the expression given in (3.17) for frames 32-230, with $\alpha = 0.1$. The second column shows the standard deviation of the values $d_{CB}(t)$ for different values of L . In order to observe the sensitivity of the measure $d_{CB}(t)$ to shifts in the correct segmentation masks, we randomly shifted the ground-truth segmentation masks with ± 10 pixels, in an attempt to simulate incorrect segmentation. The values of $D_{CB,\alpha}$, and standard deviation of $d_{CB}(t)$ using shifted segmentation maps are given in Table 3.2. The ratios of shifted values to unshifted values show that the maximum increase occurs when $L = 2$ and $L = 3$, respectively. Although the standard deviation of $d_{CB}(t)$ depend on the characteristics of the background and object color, we expect it to be small if the object is correctly segmented and the background surrounding the object is not cluttered.

3.2.4.2. Experiments with Color Histogram Differences. The results for the measure based on histogram differences are summarized in Table 3.3. We can observe that χ^2 is the most sensitive measure to the shifts in the segmentation map since the relative increase in $D_{CH,\alpha}$, and the standard deviation of $d_{CH}(t)$ are largest for the χ^2 measure when the segmentation map is shifted. This indicates that it is able to discriminate the imperfections in the segmentation/tracking results better than the other three measures. The standard deviation of $d_{CH}(t)$ is ideally expected to be low when the segmentation masks are correctly located since the color histogram of the object is not expected to change much between frames. In Figure 3.10, a plot of the χ^2 measure is given calculated using unshifted segmentation masks up to frame 100 and with masks shifted by ± 10 pixels for frames 101-230. As seen in the figure, the histogram difference measure based on χ^2 distance calculation signals the incorrectness of the segmentation mask successfully.

3.2.4.3. Experiments with the Motion Differences . Forward and backward motion estimation between successive frames of the Hall monitor sequence are performed using a hierarchical version of the Lucas-Kanade motion estimation algorithm [96]. In the first two columns of Table 3.4, the values of $D_{M,\alpha}$ and the variance of $d_M(t)$ are given

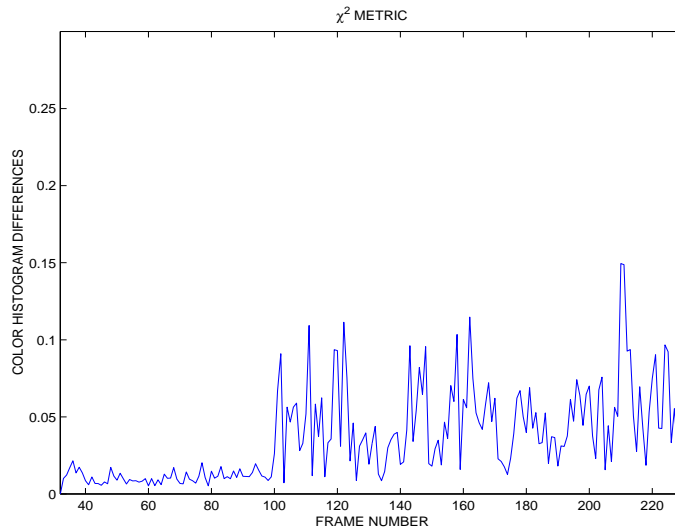


Figure 3.10. The color histogram differences between H_t and $H_{t,av}$, calculated with χ^2 measure, where segmentation maps are shifted by ± 10 pixels, starting from frame 100 for different L values. The last two columns show the ratio of shifted and unshifted values of $D_{M,\alpha}$ and standard deviation of $d_M(t)$. Since the background is stationary the $d_M(t; i)$ values decrease causing an increase in the $d_M(t)$ values.

Table 3.2. The scores for color difference measure along the object boundary

L	No Shift		± 10 pixel shift		Ratio(shifted/unshifted)	
	$D_{CB,\alpha}$ (mean)	$\text{std}(d_{CB}(t))$ (stand. dev.)	$D_{CB,\alpha}$ (mean)	$\text{std}(d_{CB}(t))$ (stand. dev.)	$D_{CB,\alpha}$ (mean)	$\text{std}(d_{CB}(t))$ (stand. dev.)
5	0.86619	0.02516	0.91175	0.03095	1.0526	1.230
4	0.86255	0.02421	0.92064	0.02819	1.0674	1.165
3	0.85781	0.02097	0.93276	0.02587	1.0874	1.234
2	0.86457	0.02194	0.93944	0.02782	1.0866	1.268

3.2.4.4. Localization of Incorrect Segmentation. In Figure 3.11 (a), we show the video object plane for the 134th frame of the Hall monitor sequence (downloaded from the web page of COST 211 group). As observed, the boundary of the object is located incorrectly except for a short segment around the shirt. The points where the color and motion differences are calculated are shown with green dots. For this example the values $\gamma_1 = 1, \gamma_2 = 0, Th = 0.64$ are used. In Figure 3.11 (b), the points of incorrectly

Table 3.3. The scores for color histogram difference measure

	No Shift		± 10 pixel shift		Ratio(shifted/unshifted)	
	$D_{CH,\alpha}$ (mean)	$\text{std}(d_{CH}(t))$ (stand. dev.)	$D_{CH,\alpha}$ (mean)	$\text{std}(d_{CH}(t))$ (stand. dev.)	$D_{CH,\alpha}$ (mean)	$\text{std}(d_{CH}(t))$ (stand. dev.)
χ^2	0.01298	0.004940	0.04065	0.025793	3.1312	5.2218
L_2	0.11711	0.035836	0.19013	0.064258	1.6236	1.7931
L_1	0.07995	0.019021	0.13943	0.047485	1.7439	2.4965
HI	0.07103	0.019882	0.13076	0.048051	1.8409	2.4168

Table 3.4. The scores for motion difference measure along the object boundary

	No Shift		± 10 pixel shift		Ratio(shifted/unshifted)	
	$D_{M,\alpha}$ (mean)	$\text{std}(d_M(t))$ (stand. dev.)	$D_{M,\alpha}$ (mean)	$\text{std}(d_M(t))$ (stand. dev.)	$D_{M,\alpha}$ (mean)	$\text{std}(d_M(t))$ (stand. dev.)
7	0.6760	0.1735	0.7338	0.1597	1.0855	0.92075
5	0.7113	0.1632	0.7704	0.1403	1.0831	0.85967

located boundary segments are marked with boxes and the correctly located segments are left unmarked. The measure (3.35) is able to support the subjective observations.

In Figure 3.13 (a), we show a segmentation of the 123th frame of the Bream sequence. As observed, the boundary of the object is located correctly except for a short segment at the bottom of the fish. The points where the color and motion differences are calculated are shown with green dots. In Figure 3.12, all the values of $d_{CB}(t; i)$ are shown along the boundary. In Figure 3.13 (b), the results using only the color difference measure with $\gamma_1 = 1, \gamma_2 = 0, Th = 0.49$ values are shown. There is a false alarm region around the dark red part.

In Figure 3.13 (c), the results using only the motion difference measure with $\gamma_1 = 0, \gamma_2 = 1, Th = 0.71$ values are shown. There are several false alarms. For the final spatial localization if incorrectly segmented regions we perform a binary AND

operation on the images (a) and (b) and obtain the result shown in Figure 3.13 (d). Except for a single point at the top, all the incorrectly segmented regions are correctly identified.

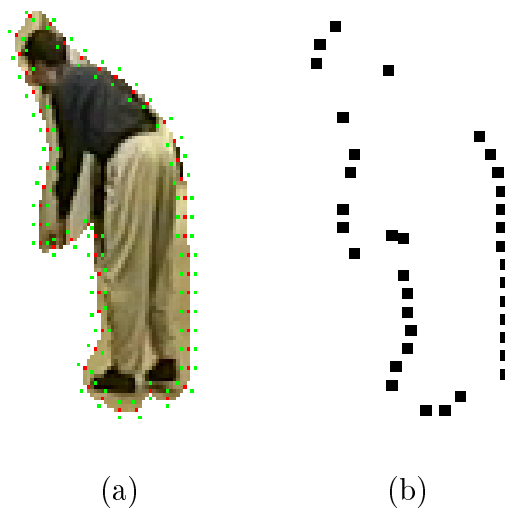


Figure 3.11. (a) The video object plane for the 134th frame (b) Incorrectly segmented regions are marked with black boxes

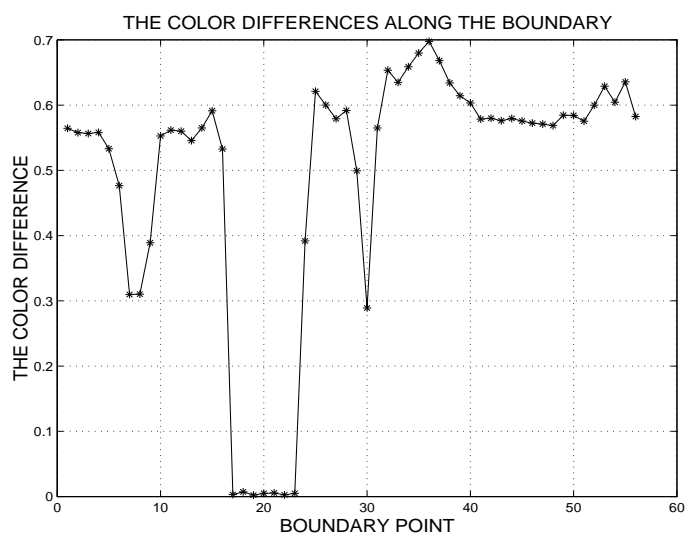
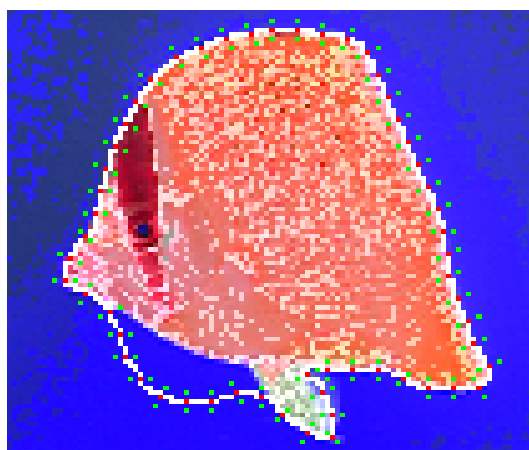
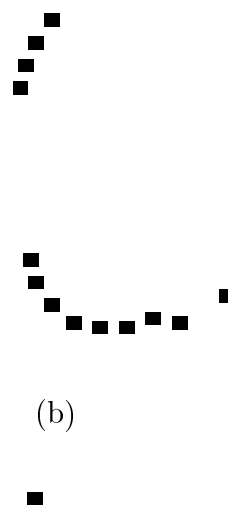


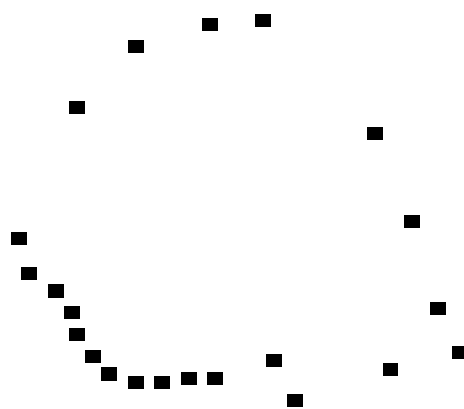
Figure 3.12. The color differences $d_{CB}(t; i)$ along the boundary of segmented Bream object



(a)



(b)



(c)



(d)

Figure 3.13. (a) Segmentation of the 123th frame (b) Incorrectly segmented regions using the color difference measure (c) Final decision of incorrectly segmented regions using the motion difference measure (d) Final decision of incorrectly segmented regions

3.3. Comparing the Ground-Truth and Non-Ground-Truth Measures

In this section, we compare the evaluation results of the performance measures with ground truth and without ground truth in order to justify the useful utilization of the performance evaluation measures when the ground-truth measures are not available.

Table 3.5 shows the performance evaluation measures of “Bream” sequence averaged for frames 100-130, for four different tracking algorithms, namely the ‘Open Loop’, ‘Edges Only’, ‘Equal Weight’ and ‘Adaptive Weight’ methods. These results are obtained from intermediate steps of the algorithm which will be described in detail in Chapter 4. The first three columns of Table 3.5 shows the results using ground truth performance measures, namely the misclassification penalty (\overline{DP}), shape penalty (\overline{DS}) and the motion penalty (\overline{DM}). The last six columns of Table 3.5, show the results for the performance evaluation measures without ground-truth: D_{CB} , is the measure of color differences along the object boundary; $D_{CH_{L_1}}$, $D_{CH_{L_2}}$, $D_{CH_{\chi^2}}$, D_{CHI} , are the measures of inter-frame histogram differences using L_1 , L_2 , χ^2 and the HI measures, respectively; D_M is the measure of motion differences along the object boundary. In Table 3.6, the standard deviations of the performance measures are given.

It can be observed that, when the ground-truth measures are high, the non-ground-truth measures also become high, and vice versa. This shows that, as expected, all the non-ground-truth measures are positively correlated with the ground-truth measures. In Section 3.4, we will analyze the amount of correlation quantitatively.

Table 3.5. The mean of performance evaluation scores for each frame of the “Bream” sequence

	Ground-truth			No Ground Truth					
	\overline{DP}	\overline{DS}	\overline{DM}	D_{CB}	$D_{CH_{L_1}}$	$D_{CH_{L_2}}$	$D_{CH_{\chi_2}}$	$D_{CH_{HI}}$	D_M
	10^{-2}	10^{-3}	10^{-1}	10^{-1}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-1}
Open Loop	5.15	2.8	2.006	7.9491	11.856	14.379	0.729	8.206	5.835
Edges Only	1.17	2.5	0.587	6.7503	8.890	11.706	0.411	5.581	5.158
Equal Weight	0.89	2.3	0.347	6.5923	8.350	11.476	0.310	5.425	5.124
Adapt. Weight	0.87	2.1	0.343	6.5915	8.480	11.641	0.335	5.310	5.061

Table 3.6. The standard deviation of the performance evaluation scores for each frame of the “Bream” sequence. The mean values are given in Table 3.5

	Ground-truth			No Ground Truth					
	DP	DS	DM	d_{CB}	$d_{CH_{L_1}}$	$d_{CH_{L_2}}$	$d_{CH_{\chi_2}}$	$d_{CH_{HI}}$	d_M
	10^{-2}	10^{-3}	10^{-1}	10^{-1}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-1}
Open Loop	4.191	0.730	2.049	0.831	6.913	7.163	0.842	6.081	1.577
Edges Only	0.299	0.694	0.562	0.291	3.890	4.976	0.402	3.458	1.203
Equal Weight	0.336	0.654	0.341	0.098	3.411	4.863	0.157	3.361	1.236
Adapt. Weight	0.325	0.650	0.375	0.113	3.517	4.894	0.235	3.241	1.022

3.4. Statistical Analysis of Ground-Truth and Non-Ground-Truth Measures

In this section we analyze the data of performance scores obtained using ground-truth and non-ground-truth metrics from different algorithms.

Let the ground-truth performance scores obtained for the t^{th} frame, video shot or a video sequence be denoted by

$$\mathbf{x}_t = \begin{bmatrix} x_{t1} & \dots & x_{tp} \end{bmatrix}^T = \begin{bmatrix} DP(t) & DS(t) & DM(t) \end{bmatrix}^T \quad (3.36)$$

consisting of the misclassification penalty (DP), shape penalty (DS) and the motion penalty (DM) and $p = 3$. Similarly let the non-ground-truth performance scores obtained for the i^{th} frame be denoted by

$$\begin{aligned} \mathbf{y}_t &= \begin{bmatrix} y_{t1} & \dots & y_{tq} \end{bmatrix}^T \\ &= \begin{bmatrix} d_{CH_{L_1}}(t) & d_{CH_{L_2}}(t) & d_{CH_{\chi^2}}(t) & d_{CH_{HI}}(t) & d_{CB}(t) & d_M(t) \end{bmatrix}^T, \end{aligned} \quad (3.37)$$

where $q = 6$ and the first four variables of which consist of the inter-frame color histogram difference metrics calculated using the L_1 , L_2 , χ^2 and the histogram intersection methods, respectively. The fifth parameter is $y_{t5} = d_{CB}(t)$ is the metric calculated from color differences along the estimated object boundary and the final parameter is $y_{t6} = d_M(t)$ is the metric of motion differences along the estimated object boundary.

Using the above vectors the following data matrix can be constructed:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \end{bmatrix} \quad (3.38)$$

$$= \begin{bmatrix} \mathbf{x}_1 & \mathbf{y}_1 \\ \vdots & \vdots \\ \mathbf{x}_n & \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_n \end{bmatrix} \quad (3.39)$$

$$= \begin{bmatrix} x_{11} & \cdots & x_{1p} & \left| & y_{11} & \cdots & y_{1q} \\ x_{21} & \cdots & x_{2p} & \left| & y_{21} & \cdots & y_{2q} \\ \vdots & \vdots & \vdots & \left| & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{np} & \left| & y_{n1} & \cdots & y_{nq} \right. \end{bmatrix}, \quad (3.40)$$

where n is the total number of observations. For example, n can be number of frames of a sequence for which the performance metrics are computed. If different tracking results are obtained for the same frame with different algorithms, the number of observations increases accordingly. For scale independence of data, the data matrix can be standardized by subtracting the mean and dividing by the standard deviation of each component such that:

$$\hat{z}_{tj} = \frac{z_{tj} - \bar{z}_j}{\sqrt{\sigma_{jj}}}, \quad (3.41)$$

$$\bar{\mathbf{z}} = \frac{1}{n} \sum_{t=1}^n \mathbf{z}_t \quad (3.42)$$

$$\sigma_{jj} = \frac{1}{n-1} \sum_{t=1}^n (z_{tj} - \bar{z}_j)^2$$

$$\Sigma = \frac{1}{n-1} \sum_{t=1}^n (\mathbf{z} - \bar{\mathbf{z}})(\mathbf{z} - \bar{\mathbf{z}})^T, \quad (3.43)$$

where $\bar{\mathbf{z}}$ is the vector of sample mean, and Σ is the matrix of sample covariances. In the rest of this chapter, the data matrix will correspond to the standardized data variables of zero mean and unit variance. The covariance matrix of the standardized data (or the correlation coefficient matrix of the original data) can be written as:

$$\Sigma = \begin{bmatrix} \Sigma_{11}(p \times p) & \Sigma_{12}(p \times q) \\ \Sigma_{21}(q \times p) & \Sigma_{22}(q \times q) \end{bmatrix} \quad (3.44)$$

3.4.1. Principal Component Analysis of Performance Measures

A principal component analysis is a method that reveals the variance-covariance behaviour of a set of data by utilizing *linear* combinations of data [113]. Often, much of

the variability in the $p + q$ dimensional data can be accounted for by a smaller number k of the principal components. The extraction of the principal components depend only on the covariance matrix of the data. The method does not require multivariate normal assumption. However, PCA analysis on multivariate normal data may provide some useful interpretations [113].

Principal components are the linear combinations of data whose variances are as large as possible. These linear combinations can geometrically be interpreted as rotating the original coordinate system with the data variables as coordinate axes to obtain a new coordinate system the axes of which represent the direction of maximum variation.

The eigenvectors of the covariance matrix Σ give us the direction of the principal components and the corresponding eigenvalues give us the amount of variation in these directions. Let the eigenvalue-eigenvector pairs of the covariance matrix S be $(\lambda_1, \mathbf{e}_1), (\lambda_2, \mathbf{e}_2), \dots, (\lambda_{p+q}, \mathbf{e}_{p+q})$. Then the i^{th} sample principal component is given by:

$$PC_i = \mathbf{e}_i^T \mathbf{z} = e_{i1}z_1 + e_{i2}z_2 + \dots + e_{i(p+q)}z_{p+q}, \quad i = 1, \dots, p + q \quad (3.45)$$

where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{p+q}$ and \mathbf{z} is any instance of the measurements \mathbf{z}_k . For example, the vector \mathbf{z}_k is composed of the $p + q$ performance measures for the k^{th} frame of a sequence. Also,

- Sample variance $(PC_i) = \lambda_i, \quad i = 1, \dots, p + q$
- Sample covariance $(PC_i, PC_k) = 0 \quad i \neq k$
- Total (standardized) sample variance = $\text{trace}(\Sigma) = p + q = \lambda_1 + \lambda_2 + \dots + \lambda_{p+q}$.
- Proportion of sample variance due to the k^{th} principal component = $\frac{\lambda_k}{p+q}, \quad k = 1, 2, \dots, p + q$.

We carried out a principal component analysis on the data obtained from the ground-truth and non-ground-truth performance evaluation metrics using the intermediate and final tracking results of the algorithm which will be described in detail in Chapter 4.

The following data were obtained from the tracking results of the ‘Bream’ sequence only. A total of 120 data vectors

$$\mathbf{z}_t = \left[DP(t) \quad DS(t) \quad DM(t) \quad d_{CH_{L_1}}(t) \quad d_{CH_{L_2}}(t) \quad d_{CH_{\chi^2}}(t) \quad d_{CH_{HI}}(t) \quad d_{CB}(t) \quad d_M(t) \right]$$

were obtained for frames $i=100, \dots, 130$, using four different tracking strategies: open loop, closed-loop with only edge and curvature energy terms, closed loop with equal weighting of all energy terms, and closed loop with adaptive weighting results. The ‘Bream’ sequence has an uncluttered background, and the color of the fish is also easily distinguished from the background. Sample frames of this sequence are shown in Chapter 4, Figure 4.8.

The correlation coefficient matrix (covariance matrix of the standardized data) is:

$$\Sigma = \begin{bmatrix} 1.00 & 0.39 & 0.86 & 0.48 & 0.36 & 0.35 & 0.56 & \underline{0.90} & 0.43 \\ 0.39 & 1.00 & 0.41 & 0.32 & 0.27 & 0.34 & 0.35 & 0.41 & 0.18 \\ 0.86 & 0.41 & 1.00 & 0.63 & 0.52 & 0.48 & 0.67 & 0.81 & 0.50 \\ \hline 0.48 & 0.32 & 0.63 & 1.00 & 0.97 & 0.77 & 0.84 & 0.59 & 0.66 \\ 0.36 & 0.27 & 0.52 & 0.97 & 1.00 & 0.67 & 0.82 & 0.46 & 0.63 \\ 0.35 & 0.34 & 0.48 & 0.77 & 0.67 & 1.00 & 0.53 & 0.54 & 0.48 \\ 0.56 & 0.35 & 0.67 & 0.84 & 0.82 & 0.53 & 1.00 & 0.56 & 0.50 \\ 0.90 & 0.41 & 0.81 & 0.59 & 0.46 & 0.54 & 0.56 & 1.00 & 0.53 \\ 0.43 & 0.18 & 0.50 & 0.66 & 0.63 & 0.48 & 0.50 & 0.53 & 1.00 \end{bmatrix}. \quad (3.46)$$

If we analyze the first row, which quantifies the correlation between the measure $DP(t)$ (pixel misclassification penalty) and the other measures, we can see that there is a high correlation (in fact the highest) between $DP(t)$ and $d_{CB}(t)$ (0.90), which is the measure of color differences along the boundary. This fact can easily be observed from plots Figure 3.14(a) and Figure 3.15(a) and also from the scatter plot of the standardized variables in Figure 3.16 (b).

It is informative to carry out a PCA on the data matrices \mathbf{X} and \mathbf{Y} in (3.38), i.e, the measures with and without ground-truth data separately. The eigenvalues and

the corresponding eigenvectors of the correlation coefficient matrix of \mathbf{X} , which is the upper left submatrix Σ_{11} of Σ in (3.46) are:

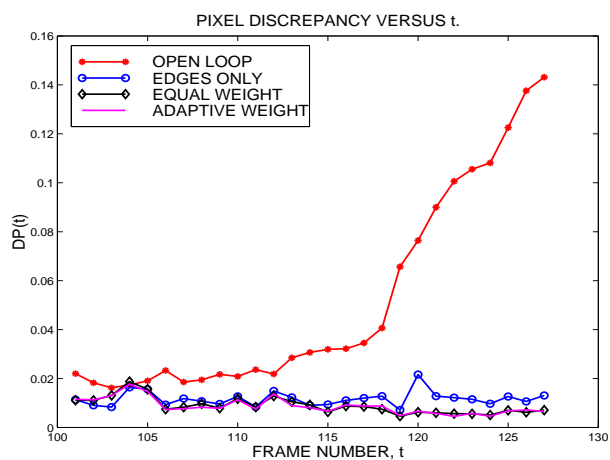
$$\begin{aligned} \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}^T &= \begin{bmatrix} 2.1362 & 0.7195 & 0.1443 \end{bmatrix}^T \\ \mathbf{e}_1 &= \begin{bmatrix} 0.6301 & 0.4453 & 0.6361 \end{bmatrix}^T \\ \mathbf{e}_2 &= \begin{bmatrix} 0.3357 & -0.8949 & 0.2939 \end{bmatrix}^T \\ \mathbf{e}_3 &= \begin{bmatrix} -0.7002 & -0.0283 & 0.7134 \end{bmatrix}^T \end{aligned}$$

The largest two eigenvalues account for 95 per cent of the variability of the data, therefore the last principal component can be disregarded reducing the size of the data vector \mathbf{x} from 3 to 2.

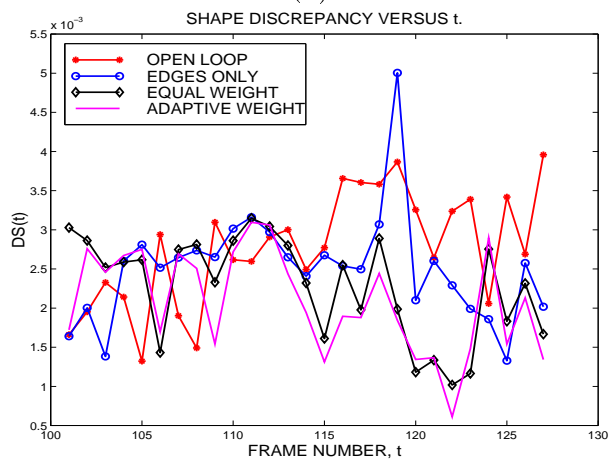
The eigenvalues and the first four corresponding eigenvectors of the correlation coefficient matrix of \mathbf{Y} , which is the lower right submatrix Σ_{22} of Σ in (3.46) are:

$$\begin{aligned} \begin{bmatrix} \lambda_1 & \dots & \lambda_6 \end{bmatrix}^T &= \begin{bmatrix} 4.22 & 0.64 & 0.52 & 0.46 & 0.12 & 0.01 \end{bmatrix}^T \\ \mathbf{e}_1 &= \begin{bmatrix} 0.47 & 0.44 & 0.38 & 0.41 & 0.34 & 0.36 \end{bmatrix}^T \\ \mathbf{e}_2 &= \begin{bmatrix} -0.22 & -0.39 & -0.004 & -0.27 & 0.75 & 0.38 \end{bmatrix}^T \\ \mathbf{e}_3 &= \begin{bmatrix} 0.014 & -0.13 & 0.56 & -0.001 & 0.31 & -0.75 \end{bmatrix}^T \\ \mathbf{e}_4 &= \begin{bmatrix} 0.030 & 0.007 & 0.63 & -0.60 & -0.37 & 0.32 \end{bmatrix}^T \end{aligned}$$

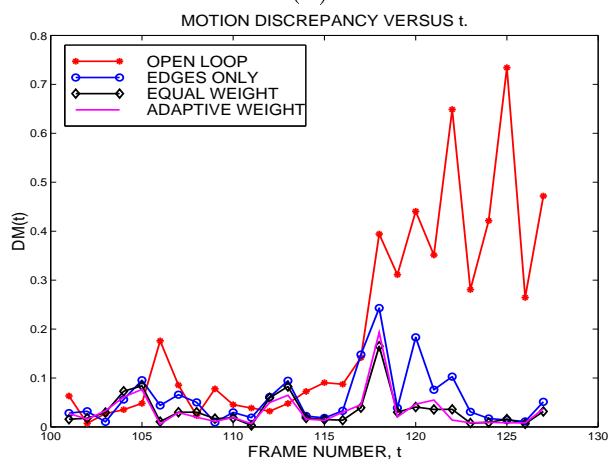
The largest four eigenvalues account for 98 percent of the variability of the data, therefore the last two principal component can be disregarded reducing the size of the data vector \mathbf{y} from 6 to 4.



(a)



(b)



(c)

Figure 3.14. Measures with ground-truth: (a) The misclassification penalty $DP(t)$ for each frame (b) The shape penalty $DS(t)$ (c) The motion penalty $DM(t)$

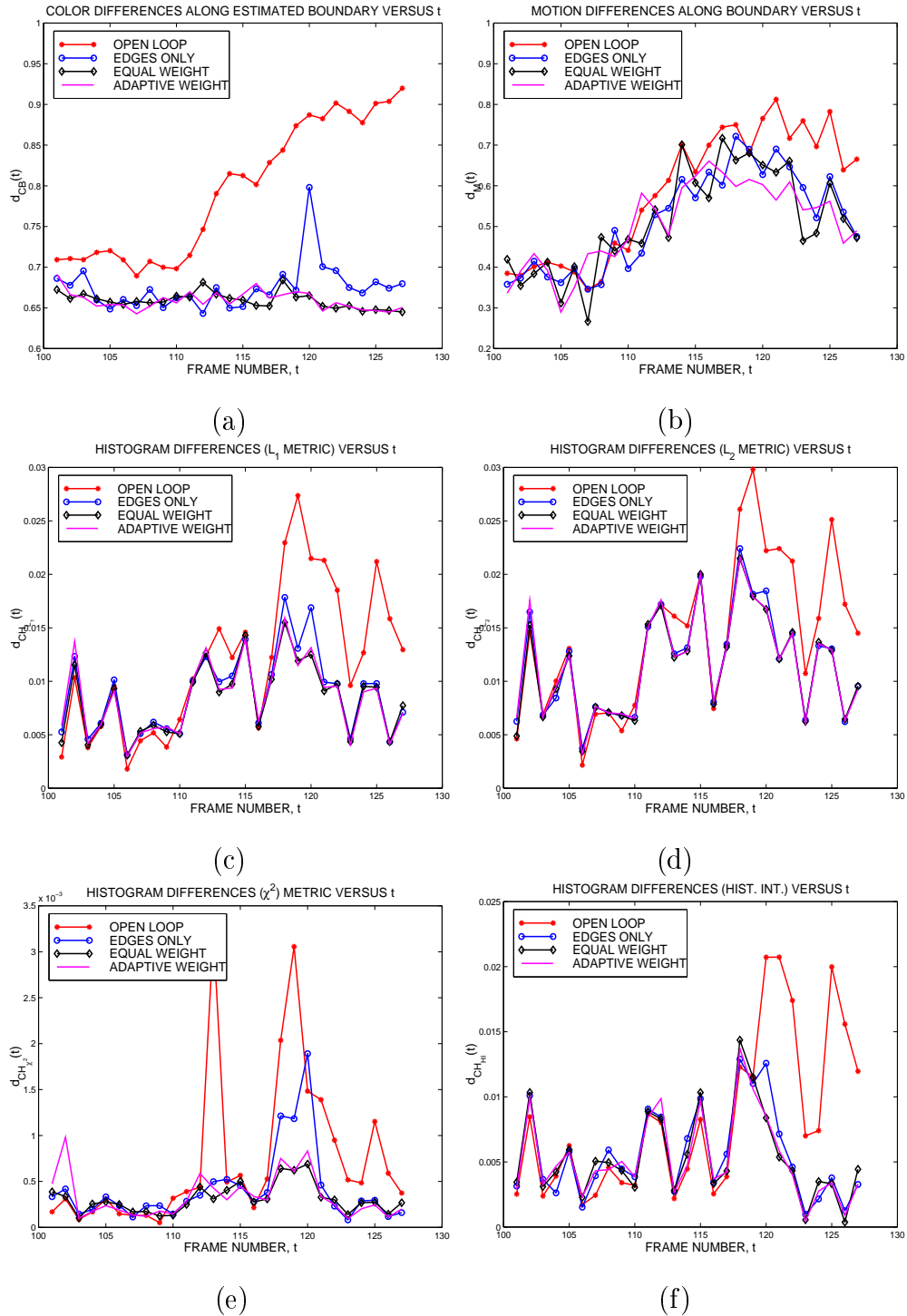


Figure 3.15. Measures without ground-truth: (a) Color differences along boundary (b) Motion differences along boundary (c), (d), (e), and (f) show the inter-frame histogram differences using L_1 , L_2 , χ^2 and the histogram intersection, respectively

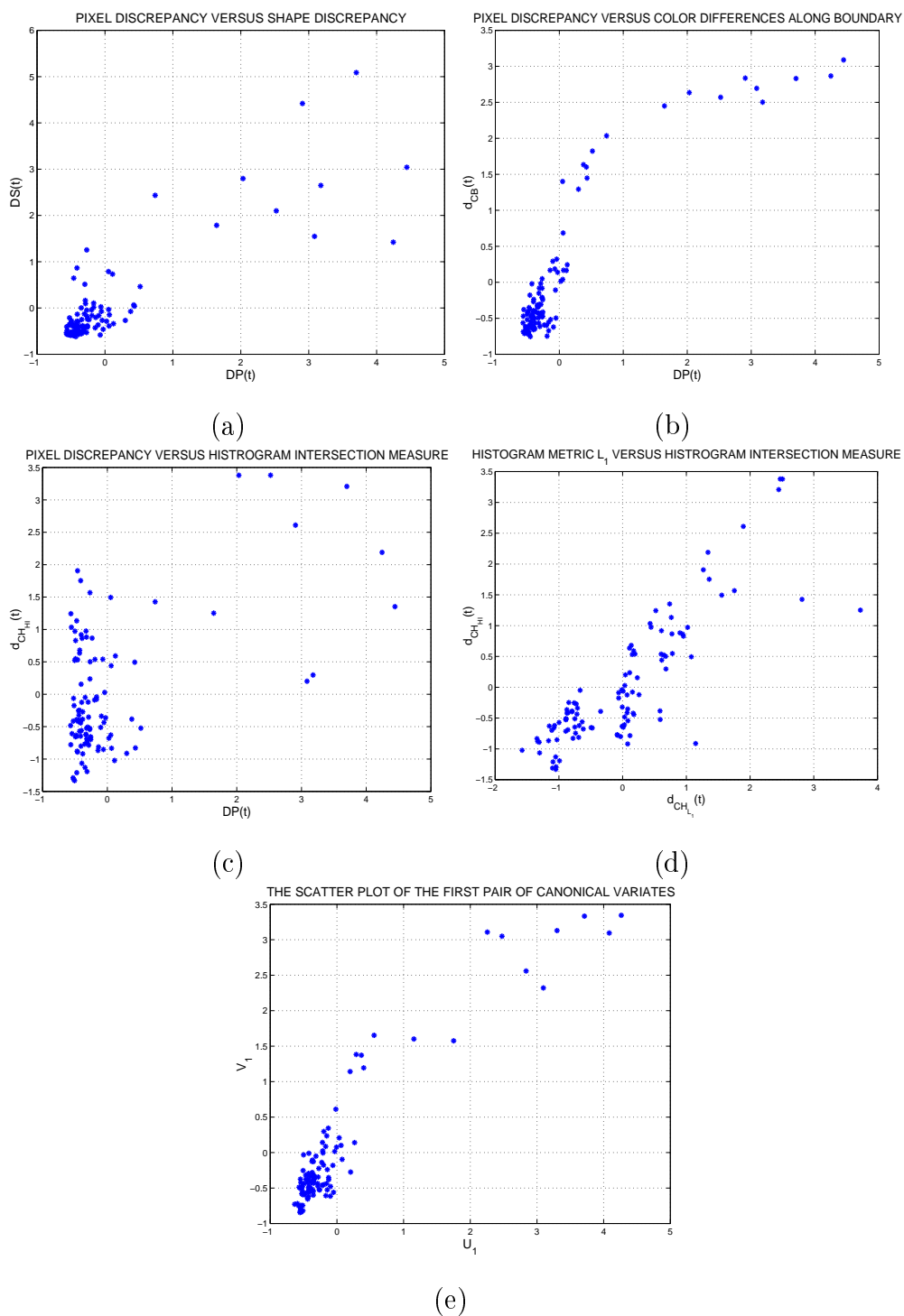


Figure 3.16. (a) Scatter plot of $DP(t)$ versus $DS(t)$ (b) $DP(t)$ versus $d_{CB}(t)$ (c) $DP(t)$ versus $d_{CH_{HI}}(t)$ (d) $d_{CH_{L_1}}(t)$ versus $d_{CH_{HI}}(t)$ (e) The scatter plot of the first pair of canonical variables

3.4.2. Canonical Correlation Analysis of Performance Measures

The association between two data sets can be quantified by using canonical correlation analysis. The idea of canonical correlation analysis is to maximize the correlation between a linear combination of one set of variables and a linear combination of the other set of variables. Then, a second pair of linear combinations of the two sets such that they have the next largest correlation and are uncorrelated with the first pair is found, and so on [113]. The pairs of linear combinations are referred to as *canonical variables* and their correlations are referred to as *canonical correlations*.

If $\text{Cov}(\mathbf{X}) = \Sigma_{11}$ ($p \times p$), $\text{Cov}(\mathbf{Y}) = \Sigma_{22}$ ($q \times q$), $\text{Cov}(\mathbf{X}, \mathbf{Y}) = \Sigma_{12}$ ($p \times q$), where Σ has full rank, then the pair of coefficient vectors \mathbf{a} ($p \times 1$) and \mathbf{b} ($q \times 1$) gives us the first pair of canonical variables $U_1 = \mathbf{a}_1^T \mathbf{X}$, $V_1 = \mathbf{b}_1^T \mathbf{Y}$ such that

$$\max_{\mathbf{a}_1, \mathbf{b}_1} \text{Corr}(U_1, V_1) = \rho_1, \quad (3.47)$$

which is achieved by

$$U_1 = \underbrace{\mathbf{e}_1^T \Sigma_{11}^{-1/2}}_{\mathbf{a}_1} \mathbf{X} \quad (3.48)$$

$$V_1 = \underbrace{\mathbf{f}_1^T \Sigma_{22}^{-1/2}}_{\mathbf{b}_1} \mathbf{Y} \quad (3.49)$$

The k^{th} pair of canonical variates $k = 2, 3, \dots, p$,

$$U_k = \underbrace{\mathbf{e}_k^T \Sigma_{11}^{-1/2}}_{\mathbf{a}_k} \mathbf{X} \quad (3.50)$$

$$V_k = \underbrace{\mathbf{f}_k^T \Sigma_{22}^{-1/2}}_{\mathbf{b}_k} \mathbf{Y} \quad (3.51)$$

maximizes

$$\max_{\mathbf{a}_k, \mathbf{b}_k} \text{Corr}(U_k, V_k) = \rho_k, \quad (3.52)$$

among those linear combinations that are uncorrelated with the previous $1, \dots, k-1$ variables. The parameters $\rho_1^2 \geq \rho_2^2 \geq \dots \rho_p^2$ are the eigenvalues of the matrix

$$\Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} \Sigma_{11}^{-1/2} \quad (3.53)$$

and $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_p$ ($p \times 1$) are the corresponding eigenvectors. The parameters $\rho_1^2 \geq \rho_2^2 \geq \dots \rho_p^2$ are also the p largest ($p \leq q$) eigenvalues of the matrix

$$\Sigma_{22}^{-1/2} \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1/2} \quad (3.54)$$

with corresponding ($q \times 1$) eigenvectors $\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_p$.

Some properties of the canonical variates are:

$$\text{Var}(U_k) = \text{Var}(V_k) = 1 \quad (3.55)$$

$$\text{Cov}(U_k, U_l) = \text{Corr}(U_k, U_l) = 0, \quad k \neq l \quad (3.56)$$

$$\text{Cov}(V_k, V_l) = \text{Corr}(V_k, V_l) = 0, \quad k \neq l \quad (3.57)$$

$$\text{Cov}(U_k, V_l) = \text{Corr}(U_k, V_l) = 0, \quad k \neq l \quad (3.58)$$

In order to find the canonical variates of the data described by the covariance matrix given in (3.46), we first find the eigenvalues and eigenvectors of the matrices (3.53) and (3.54). The eigenvalues of the matrix (3.53) are:

$$\begin{bmatrix} \rho_1^2 & \rho_2^2 & \rho_3^2 \end{bmatrix} = \begin{bmatrix} 0.8750 & 0.2417 & 0.0494 \end{bmatrix}. \quad (3.59)$$

The first pair of linear transformation parameters corresponding to the largest eigenvalue is:

$$\begin{aligned} \mathbf{a}_1 &= \begin{bmatrix} 0.8004 & 0.0266 & 0.2133 \end{bmatrix}^T \\ \mathbf{b}_1 &= \begin{bmatrix} 0.6107 & -0.5561 & -0.2803 & 0.2409 & 0.8995 & -0.0309 \end{bmatrix}^T \end{aligned} \quad (3.60)$$

These linear transformation coefficients tell us to take 80 percent of DP (misclassification penalty) together with 20 percent of DM (motion penalty) among the set of ground-truth metrics and 90 percent of the d_{CB} measure (color differences along boundary) among the set of non-ground-truth metrics, so that the two set of performance metrics will be maximally correlated. The maximum correlation between the first canonical variate pair is:

$$\text{Corr}(U_1, V_1) = \rho_1 = 0.9354. \quad (3.61)$$

The scatter plot of the first two principal components c_1, c_2 is given in Figure 3.16(f).

A geometrical interpretation of the canonical correlation analysis is possible. The linear transformation matrices consisting of the \mathbf{a}_i and \mathbf{b}_i vectors can be interpreted as rotation matrices [113]. Therefore, by rotating the axes of the two set of variables, we try to minimize the angles between their axes. The maximum correlation value in (3.61) can be interpreted as the cosine of the minimum angle i.e, $\cos^{-1}(0.9354) = 20$ degrees.

The second pair of linear transformations are:

$$\begin{aligned} \mathbf{a}_2 &= \begin{bmatrix} 1.7515 & -0.2081 & -1.8169 \end{bmatrix}^T \\ \mathbf{b}_2 &= \begin{bmatrix} 1.3960 & -0.9595 & 0.4048 & 0.5259 & -0.9781 & 0.1516 \end{bmatrix}^T \end{aligned} \quad (3.62)$$

The correlation between the second pair of canonical variates is:

$$\text{Corr}(U_2, V_2) = \rho_2 = 0.4916. \quad (3.63)$$

Finally, the third pair of linear transformations are:

$$\mathbf{a}_3 = \begin{bmatrix} -0.1800 & -1.0805 & 0.7005 \end{bmatrix}^T \quad (3.64)$$

$$\mathbf{b}_3 = \begin{bmatrix} 4.5686 & -2.9280 & -1.6525 & -0.5208 & -0.5679 & 0.5283 \end{bmatrix}^T$$

The correlation between the second pair of canonical variates is:

$$\text{Corr}(U_3, V_3) = \rho_2 = 0.2223. \quad (3.65)$$

4. VIDEO OBJECT TRACKING WITH FEEDBACK OF PERFORMANCE MEASURES

The problem of 2-D object tracking has attracted much attention due to its many applications in computer vision and video processing, including surveillance, content-based indexing and retrieval, object-based video coding, and video post-production. Some of these applications such as surveillance, require automatic real-time processing while tolerating some performance inaccuracy. In other applications, such as video-post processing and object-based coding, accuracy is very important while the processing may not need to be done in real-time and a reasonable amount of user interaction is allowed (in fact even desired). Thus, it is of interest to develop a generic scalable video object tracking framework that can address all of these diverse requirements.

In this chapter, we propose a scalable video object tracking framework, which employs the performance evaluation measures introduced in Section 3.2 in a feedback loop [114]. An overview of the framework is given in Section 4.1. In the (near) real-time mode, our scheme employs an open-loop contour tracking paradigm, which is presented in Section 4.2. In the pixel accurate tracking mode, a feedback loop based on performance measures comes into play as discussed in Section 4.3. The object contour is divided into sub-contours, and several low-level features such as color edge, color segmentation, motion models, and motion segmentation information are employed in the feedback loop to track each subcontour. Section 4.4 details the main novelty of the proposed closed-loop scheme, that is the use of feedback performance scores to automatically adjust the weight parameters in the energy minimization. Section 4.5 provides experimental results to demonstrate the performance improvements.

4.1. Overview of The Framework

We propose a scalable framework for 2-D video object tracking as shown in the flowchart in Figure 4.1. The block depicted by the dashed lines on the left part of Figure 4.1 performs a coarse tracking of the location of the object from frame $t - 1$

to t in real-time or near real-time. The global motion compensation box is activated in the presence of fast camera motion, such as zoom or pan. If a feature point cannot be properly tracked due to occlusion, temporal and spatial prediction schemes are employed for that point until (and if) it becomes uncovered again. We note that this coarse boundary tracking block may be replaced by other real-time region (blob) tracking schemes without affecting the rest of the framework.

However, the output of the coarse tracker does not always yield pixel accurate contour localization. Thus, the remainder of the proposed framework, which is indicated by dashed lines on the right of Figure 4.1, deals with enhancing the accuracy of the tracked contour (possibly for non real-time applications) by using a closed-loop energy-minimization scheme. The boundary is modeled as a union of contour segments that minimize a set of energy terms derived from edges [115], color segmentation boundaries and motion segmentation boundaries [28]. After each iteration, the goodness of the tracking results are evaluated using a set of performance measures (that do not need ground-truth information) to fine tune the weights assigned to each low-level cue, such as color edges, color segments, and motion segments.

The advantages of the tracking framework can be listed as follows:

- It can track non-rigid objects as well as rigid ones without prior training since no parametric shape space model is assumed.
- It is robust to abrupt changes in motion direction of the object, since we do not impose any prior motion model to the object.
- It can track moving objects in cluttered scenes successfully, since motion segmentation boundaries are taken into account.
- It is capable of tracking limbed objects composed of moving regions with different motion parameters (articulated motion).
- It is reasonably robust to occlusions.

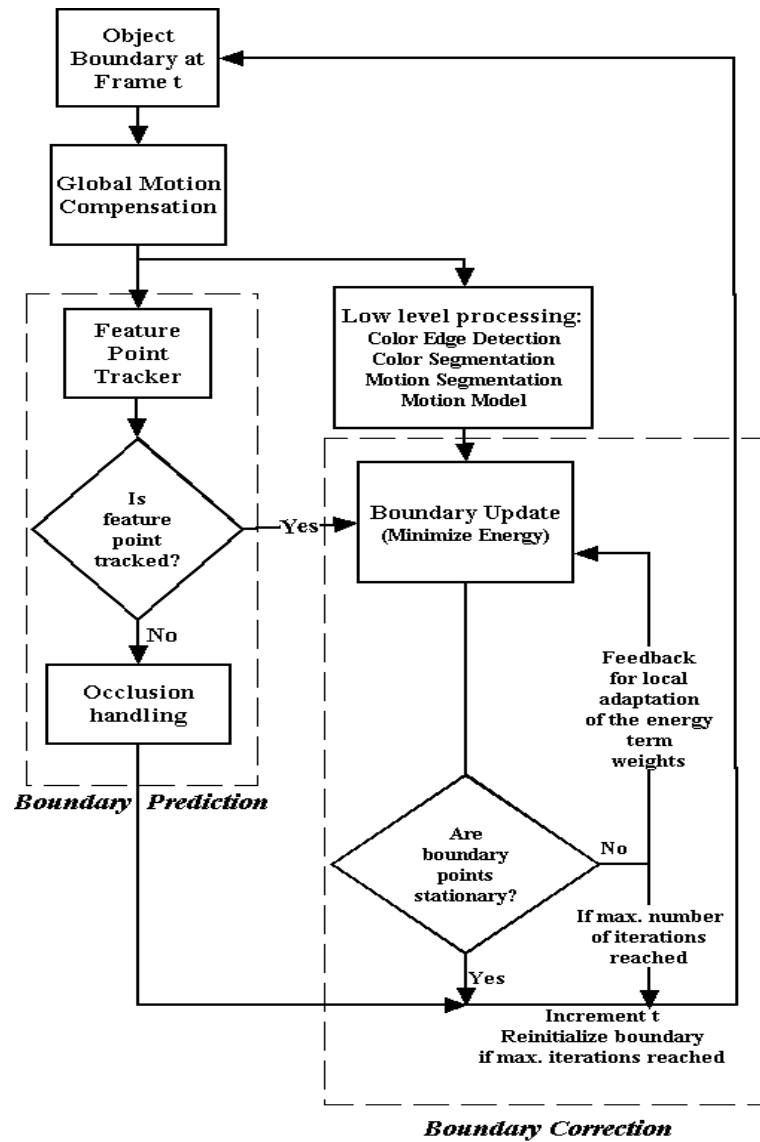


Figure 4.1. The tracking framework consists of the *boundary prediction* and the *boundary correction* steps

4.2. Open-Loop Boundary Prediction

In this section, we present a new open-loop boundary prediction method based on contour segmentation and piecewise contour mapping. We also address handling of occlusions. The open-loop part of the proposed framework can be used both in interactive (non real-time) or real-time mode.

4.2.1. Boundary Initialization

In the interactive (non real-time) mode, the object of interest may be determined in the first frame by the user. For example, the intelligent scissors [116] algorithm may be used for this purpose. In the real-time (non-interactive) mode, a change detection algorithm [5] may be used to determine the object of interest in the first frame under certain assumptions.

4.2.2. Boundary Prediction

In the following, we present our new boundary prediction method, where the Shi-Tomasi feature tracking algorithm [26] has been used in one of the steps. The main idea of our algorithm is also illustrated in Figure 4.2.

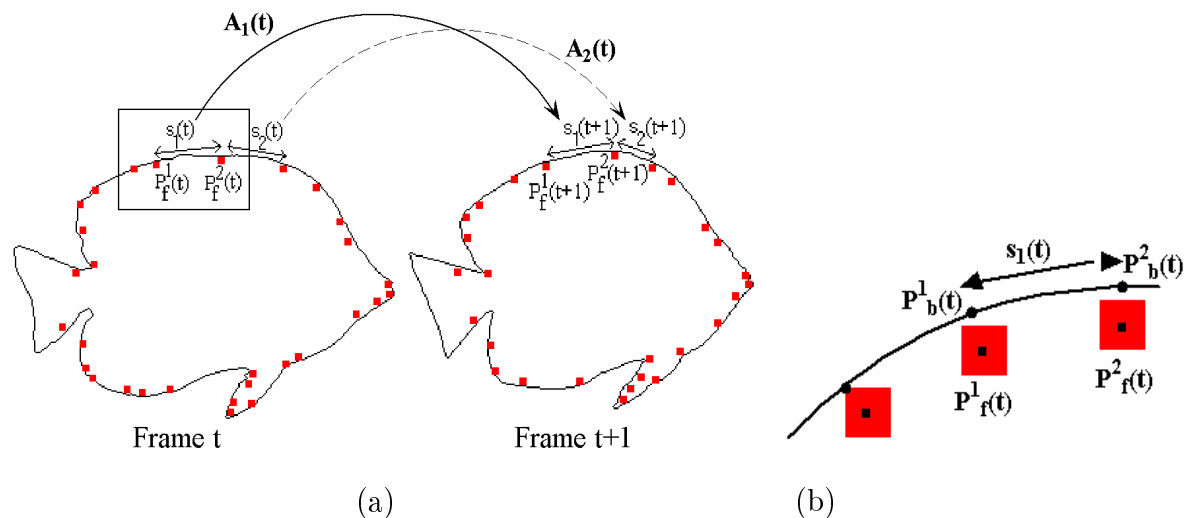


Figure 4.2. (a) Locally adaptive boundary prediction using feature point tracking (b) A magnified view of the section of the boundary shown in the box in (a)

- I. Let the boundary of the object (the segmentation map) in the first frame be given, and represented by the $2 \times N_{b,t}$ matrix \mathbf{B}^t consisting of all boundary pixels at frame t , where

$$\mathbf{B}(t) = \left[\mathbf{p}_b^1(t) \quad \mathbf{p}_b^2(t) \quad \dots \quad \mathbf{p}_b^{N_b(t)}(t) \right]^T, \quad (4.1)$$

$$\mathbf{p}_b^i(t) = \begin{bmatrix} x_b^i(t) & y_b^i(t) \end{bmatrix}^T, \quad i = 1, \dots, N_b(t), \quad (4.2)$$

The variables $(x_b^i(t), y_b^i(t))$ denote the location of the i^{th} boundary pixel at frame t .

We first find a given number $N_f(t)$ of “good” feature points [26] inside the segmentation map that are *close* to the boundary, which are shown by the red squares in Figure 4.2 (a) and (b). A feature point is close to the boundary if its minimum distance from the boundary is smaller than a certain threshold (3-5 pixels). Let $\mathbf{F}(t)$ denote the $2 \times N_f(t)$ matrix of detected feature points, which are arranged in their order of appearance while tracing the boundary:

$$\mathbf{F}(t) = \begin{bmatrix} \mathbf{p}_f^1(t) & \mathbf{p}_f^2(t) & \dots & \mathbf{p}_f^{N_f(t)}(t) \end{bmatrix}^T, \quad (4.3)$$

$$\mathbf{p}_f^i(t) = \begin{bmatrix} x_f^i(t) & y_f^i(t) \end{bmatrix}^T, \quad i = 1, \dots, N_f(t). \quad (4.4)$$

II. Segment the boundary contour by “projecting” each feature point onto the boundary. First, for each feature point $\mathbf{p}_f^i(t), i = 1, \dots, N_f(t)$, we find the nearest boundary pixel $\mathbf{p}_b^j(t), j \in \{1, \dots, N_b(t)\}$ such that,

$$\begin{aligned} \|\mathbf{p}_f^i(t) - \mathbf{p}_b^j(t)\| &< \|\mathbf{p}_f^i(t) - \mathbf{p}_b^k(t)\|, \quad i = 1, \dots, N_f(t), \\ &\forall j, k \in \{1, \dots, N_b(t)\}, j \neq k. \end{aligned} \quad (4.5)$$

The projection points are shown by black circles in Figure 4.2 (b). Then, the boundary is divided into $M_t + 1$ segments using the projection points found in (4.5). Therefore, a segmented representation of the contour of the object at time t is given by

$$\mathbf{B}(t) = \begin{bmatrix} \mathbf{s}_1(t) & \mathbf{s}_2(t) & \dots & \mathbf{s}_{N_f(t)}(t) \end{bmatrix}^T,$$

where each $(2 \times q_i)$ submatrix $\mathbf{s}_i(t)$ consists of the locations of the q_i boundary points between the projections of the feature points $\mathbf{p}_f^i(t)$ and $\mathbf{p}_f^{i+1}(t)$ onto the boundary. Note that $q_1 + q_2 + \dots + q_{N_f(t)} = N_b(t)$, so that $\mathbf{B}(t)$ is $2 \times N_b(t)$. The

selected feature point pairs $(\mathbf{p}_f^i(t), \mathbf{p}_f^{i+1}(t))$ and the associated boundary segments $(\mathbf{s}_i(t))$ are illustrated in Figure 4.2.

- III. Track the selected feature points to the next frame using the algorithm in [26]. For each feature point $\mathbf{p}_f^i(t)$ at frame t , we estimate its location at frame $t + 1$, $\mathbf{p}_f^i(t + 1)$ (for the time being assume that all feature points have been successfully tracked to the next frame).
- IV. Using the motion information of each successive pair of feature points $(\mathbf{p}_f^i(t) \rightarrow \mathbf{p}_f^i(t + 1), \mathbf{p}_f^{i+1}(t) \rightarrow \mathbf{p}_f^{i+1}(t + 1))$, we estimate a transformation matrix for the boundary segment \mathbf{s}_i^t such that

$$\begin{bmatrix} x_f^i(t + 1) & y_f^i(t + 1) & 1 \\ x_f^{i+1}(t + 1) & y_f^{i+1}(t + 1) & 1 \end{bmatrix} = \begin{bmatrix} x_f^i(t) & y_f^i(t) & 1 \\ x_f^{i+1}(t) & y_f^{i+1}(t) & 1 \end{bmatrix} \mathbf{A}_i(t) \quad (4.6)$$

$$\mathbf{A}_i(t) = \begin{bmatrix} z_i(t) \cos \theta_i(t) & -z_i(t) \sin \theta_i(t) & 0 \\ z_i(t) \sin \theta_i(t) & z_i(t) \cos \theta_i(t) & 0 \\ u_i(t) & v_i(t) & 1 \end{bmatrix}, \quad (4.7)$$

where $z_i(t)$ denotes scale parameter, $\theta_i(t)$ denotes the rotation angle, and $u_i(t), v_i(t)$ denote two translation parameters for the boundary segment $\mathbf{s}_i(t)$.

- V. Transform i^{th} segment of the boundary at frame t ($\mathbf{s}_i(t)$) using the transformation matrix $\mathbf{A}_i(t)$ to obtain the predicted object boundary in frame $t + 1$.

In the interactive mode, the user has the option of declaring the whole or parts of the object as rigid, as well as selecting an appropriate motion model (such as affine or perspective) for each rigid segment of the object. In that case, feature points *inside the rigid parts of the object* may also be used in model parameter estimation to add robustness to the algorithm. This option is not available in the real-time (non-interactive) tracking mode.

4.2.3. Occlusion Handling

Sometimes, a feature point may not be properly tracked to the next frame. Note that, the feature tracker employs block-matching using a block of pixels centered about

the feature point. A feature point is declared as untracked when a certain MSE threshold is exceeded [26]. There are two main reasons for failure of tracking:

- The object region around the feature point has been occluded (self-occlusion or occlusion by another object).
- The feature point is near the boundary (that is, the block centered at the feature point contains pixels from the background) and the background pixels has changed.

If a feature block is not tracked properly, we try to predict its location using motion information of the successfully predicted nearby (temporal or spatial) features. We propose a combined spatial and temporal prediction scheme, which is described in detail in the following.

4.2.3.1. Temporal Prediction. Let the location of a feature block at frame t , which was not tracked to frame $t + 1$, be $\mathbf{p}_f^i(t), i \in \{1, \dots, N_f(t)\}$. Its location is predicted using the average of its motion vectors in the last K frames:

$$\hat{\mathbf{p}}_f^i(t + 1) = \mathbf{p}_f^i(t) + \frac{1}{K} \sum_{k=0}^{K-1} \mathbf{v}_f^i(t - k), \quad (4.8)$$

$$\mathbf{v}_f^i(t) = \mathbf{p}_f^i(t) - \mathbf{p}_f^i(t - 1) \quad (4.9)$$

where $\mathbf{v}_f^i(t)$ denotes the motion vector of feature block i at frame t , and K is the number of frames to average the motion vector. The parameter K may be adjusted depending on the activity present in the scene.

4.2.3.2. Spatial Prediction. Another approach for predicting the location of an untracked feature point at frame $t + 1$ is to use other feature points at frame t that have been tracked to frame $t + 1$ and have “similar” properties. Two feature points are “similar” if they are spatially close to each other and move coherently. In order to group feature points at frame t , based on their similarity we first form a vector for each

feature point:

$$\mathbf{c}_i^t = \left[\mathbf{p}_f^i(t) \quad \mathbf{v}_f^i(t) \right]^T, \quad (4.10)$$

where $\mathbf{v}_f^i(t)$ was defined in (4.9). Then, the vectors $\mathbf{c}_i^t, i \in \{1, \dots, N_f(t)\}$ are grouped using the k-means clustering algorithm [117]. This gives us G groups which are labeled as $\{g_1(t), g_2(t), \dots, g_G(t)\}$.

The spatial prediction for untracked features is performed as follows:

$$\hat{\mathbf{p}}_f^i(t+1) = \mathbf{p}_f^i(t) + \frac{\sum_j (\mathbf{p}_f^j(t+1) - \mathbf{p}_f^j(t))}{\sum_j 1}, \quad (4.11)$$

if feature points i and j are in the same group. The temporal and spatial prediction results are combined to estimate the feature point location.

4.2.3.3. Search Around The Predicted Location. The untracked features and their estimated locations are put in a list. In the next frame, we search around the estimated location to detect whether the feature point has become uncovered again; the matching criterion being the mean squared error. For the feature blocks that contain background pixels, the search is done in such a manner that only the pixels that fall into the segmentation map are considered in the MSE calculation. This is achieved by assigning the background pixels to a uniform value.

4.3. Closed-Loop Boundary Correction

The predicted boundary found in the previous section may not exactly match the object boundary due to inaccuracies in feature point tracking and motion estimation. However, the predicted boundary is generally close enough to the actual object boundary to allow for snake-based boundary updating. This section presents the details of the proposed boundary correction scheme, also depicted in the right inset in Figure 4.1.

The active contour model was first introduced in [118]. A recent paper [40]

presents a new approach of associating the energy terms with boundary segments rather than node points in a dynamic programming energy minimization framework [119]. If only edge-based energy terms are used, the active contour is easily distracted by background clutter or inside texture and may snap to the wrong edges. The new formulation of the energy terms introduced in this section fuses the information from the color segmentation boundaries, color edges and the motion segmentation boundaries to prevent such distractions. We associate the energy terms with boundary segments as in [40].

Our boundary representation consists of polygonal snakes. Let $\mathbf{n}_i^t = (x_i^t, y_i^t)$, $i = 1, \dots, N$ denote the N selected node points along the predicted object boundary at frame t . The superscript t will be dropped in the rest of the Chapter. Note that these points are different from the ones defined in (4.2) and (4.5). These node points are selected from the boundary points where the curvature value is high. Intermediate points are inserted if the distance between two consecutive points is large. In the rest of the chapter, the superscript t will be dropped. The node points are chosen as high curvature points along the boundary. The snake energy of the boundary is expressed as:

$$E_{snake} = \sum_{i=1}^N (E_{int,i} + E_{ext,i}) \quad (4.12)$$

$$E_{int,i} = \beta_{curv,i} E_{curv,i} \quad (4.13)$$

$$E_{ext,i} = \beta_{col,i} E_{col,i} + \beta_{edge,i} E_{edge,i} + \beta_{mot,i} E_{mot,i} \quad (4.14)$$

where $E_{col,i}$, $E_{edge,i}$ and $E_{mot,i}$ are external energy terms which are calculated from color segmentation boundaries, edges and motion segmentation boundaries, respectively. $E_{curv,i}$ is an internal energy term associated with the curvature of the boundary segment between nodes $i - 2$ and i . The parameters $\beta_{curv,i}$, $\beta_{col,i}$, $\beta_{edge,i}$ and $\beta_{mot,i}$ are the weighting coefficients of the energy terms. The energy terms are elaborated in this section, and the weighting coefficients are discussed in Section 4.4.

The snake energy is minimized using the dynamic programming method [119],

where the search locations for each node are selected along the normal lines drawn to the $\{\mathbf{n}_i\}$ node points as illustrated in Figure 4.4. The dynamic programming step is iterated until most of the boundary points (e.g. %95) are stationary. In the following we discuss the external and internal energy terms in detail.

4.3.1. External Energy Terms

A valid assumption in object segmentation and tracking is that object boundaries coincide with color boundaries. However, color segmentation boundaries and edges contain different information as can be observed in Figure 4.3. If the object is also moving, object boundaries are also expected to coincide with motion boundaries. Based on these assumptions, we present new external energy terms below.

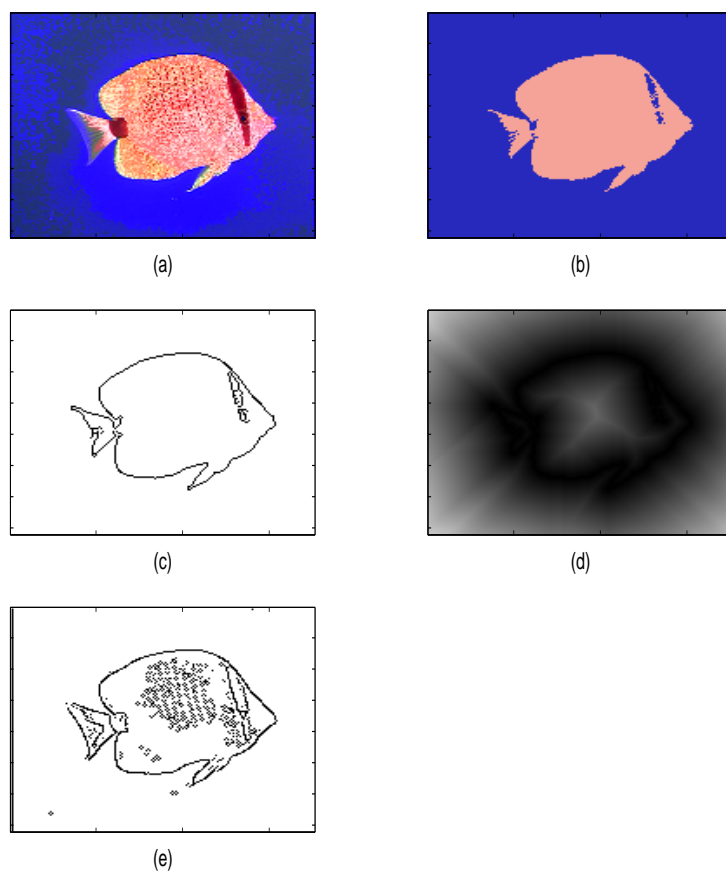


Figure 4.3. (a) The first frame of “Bream” (b) Two class color segmentation (c) Color segmentation boundaries (d) The Chamfer 3-4 transform of color segmentation boundaries (e) The edges

4.3.1.1. Color Boundary Energy. The color segmentation of each frame is estimated using the fuzzy c-means algorithm [83], where the number of classes are determined by the user. Alternatively, the number of classes can also be selected using cluster validity approaches such as Zhang et. al. [120] approach. Then, the color boundaries are extracted from the segmented frame to obtain contours (see Figure 4.3(c)). This contour map should be transformed such that it will be able to guide the snake nodes to the correct locations, even from distant places. This is achieved by using the Chamfer 3-4 distance transformation [121]. This transform approximates the Euclidean distance of a pixel location to the nearest contour point by transforming the binary contour image using the mask:

$$\begin{bmatrix} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{bmatrix}. \quad (4.15)$$

In other words, a pixel location that is in the 4-neighborhood of a boundary pixel will have a value of 3, and a pixel which is a diagonal neighbor will have a value of 4. In Figure 4.3(d), the Chamfer distance transformation of the color segmentation boundaries and their small variation are illustrated. If the Chamfer distance transform of the color segmentation is denoted by $CC(x, y)$, then the color boundary energy term is defined as follows:

$$E_{col,i} = \frac{\sum_{(x,y) \in \overline{\mathbf{n}_i(l)\mathbf{n}_{i-1}(k)}} CC(x, y)}{\|\mathbf{n}_i(l) - \mathbf{n}_{i-1}(k)\|}, \quad k = 1, \dots, N_s; l = 1, \dots, N_s. \quad (4.16)$$

In (4.16), $\overline{\mathbf{n}_i(l)\mathbf{n}_{i-1}(k)}$ denotes the line segment that is connecting the current search nodes $\mathbf{n}_{i-1}(k)$ and $\mathbf{n}_i(l)$ (see Figure 4.4 and Figure 4.5) during energy minimization using dynamic programming [119]. The indices k and l denote the search locations along the normal lines (which are the angle bisectors) drawn at nodes $i - 1$ and i , respectively and N_s is the total number of search locations at a given node.

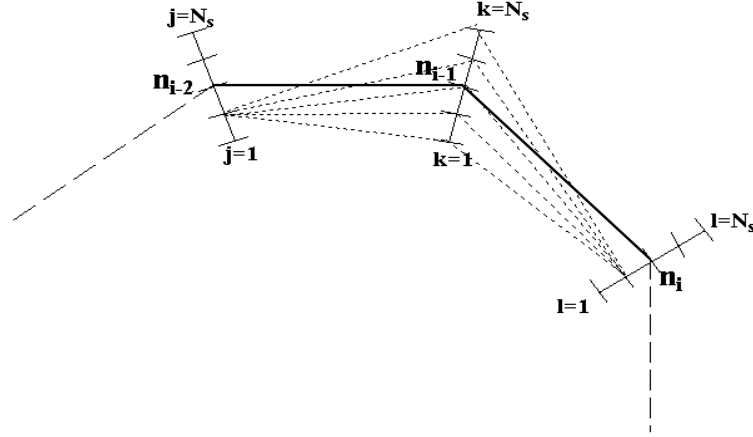


Figure 4.4. Illustration of search locations at each node during dynamic programming

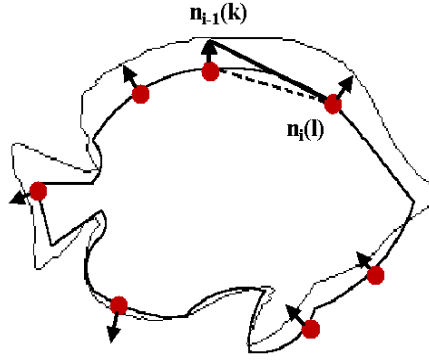


Figure 4.5. Illustration of energy minimization using dynamic programming. The correct and estimated (to be updated) object boundaries are shown by the thin and thick lines, respectively

4.3.1.2. Edge Boundary Energy. Edges are found using the Canny edge detector [115] as seen in Figure 4.3(e). Note that the edge map is quite noisy compared to the color segmentation boundaries. Rather than finding the image gradient values as in [40], we use the Chamfer distance transformation on the Canny edge field to obtain the terrain that the snake will slide on. This approach gives us a smoothly varying field that enables the snake to move towards the edges even if the image gradient is zero at its current location. This enables the snake to be attracted to the edges from large distances. Let the Chamfer distance transform of the edge map be denoted as $CE(x, y)$. The edge energy of a searched line segment is found as follows:

$$E_{edge,i} = \frac{\sum_{(x,y) \in \overline{\mathbf{n}_i(l)\mathbf{n}_{i-1}(k)}} CE(x, y)}{\|\mathbf{n}_i(l) - \mathbf{n}_{i-1}(k)\|}, \quad k = 1, \dots, N_s; \quad l = 1, \dots, N_s. \quad (4.17)$$

4.3.1.3. Motion Boundary Energy. When the object is moving with a motion pattern different from the background, the motion boundaries provide an important cue about the object boundaries. However, most dense motion estimation approaches such as block matching and Lucas-Kanade method [94], give poor motion boundaries since they tend to smooth the motion field by using motion constancy assumptions inside blocks. Therefore, methods to refine the motion segmentation boundaries using the more reliable color boundaries has been suggested [28] to perform reliable motion segmentation. In this section, we introduce a new snake energy term that utilizes this important cue: color refined motion segmentation boundaries.

The dense motion estimation is performed by the hierarchical version of Lucas-Kanade motion estimation algorithm [122]. Then, following the approach of [28], the dense motion vectors are clustered into affine motion classes using the k-means algorithm. Then, color patches of the current frame obtained by color segmentation are assigned to one of these motion classes by comparing the dense motion field of the color patch with the pre-computed motion cluster centroids. For details the reader is referred to [28].

The motion segmentation boundaries are also transformed using the Chamfer 3-4 transform resulting in the $CM(x, y)$ field. The external energy term of motion segmentation boundaries is calculated as:

$$E_{mot,i} = \frac{\sum_{(x,y) \in \overline{\mathbf{n}_i(l)\mathbf{n}_{i-1}(k)}} CM(x, y)}{\|\mathbf{n}_i(l) - \mathbf{n}_{i-1}(k)\|}, \quad k = 1, \dots, N_s; l = 1, \dots, N_s. \quad (4.18)$$

where $CM(x, y)$ denotes the Chamfer distance transform value of the color-refined motion segmentation boundary at location (x, y) .

4.3.2. Internal Energy Term

The curvature energy term is estimated using the following equation [40]:

$$E_{curv,i} = 2 + 2 \cos(\angle \overline{\mathbf{n}_{i-1}(k)\mathbf{n}_{i-2}(j)}, \overline{\mathbf{n}_{i-1}(k)\mathbf{n}_i(l)}), \quad (4.19)$$

where the indices j, k and l denote the search locations along the normal lines drawn from the nodes $i - 2, i - 1$ and i , respectively as shown in Figure 4.4.

4.3.3. Selection of Search Directions

In [40], the search locations during dynamic programming are chosen as uniformly located points along the normal lines (angle bisectors) drawn at boundary nodes. However, a more efficient approach is to use the knowledge of the object and background colors obtained from the tracking results of the previous frames. As illustrated in Figure 4.5, the search direction (towards the inside of the segmentation map or towards outside) can be pre-selected at each node. If the color of a search node is an element of the set of object colors, then we only search towards outside of the estimated boundary, otherwise we search towards the inside on the normal line.

In order to make this decision, we first estimate the probability density functions for the object and background colors using the tracking results of the previous frame. This is done by estimating the normalized color histograms. Let $H_o(c)$ and $H_b(c)$ denote the PDF's of the object and background colors estimated using the color histograms, where c denotes the color. Let $I(\mathbf{n}_i(t))$ be an indicator function for the location of the current boundary node \mathbf{n}_i^t (inside or outside the actual object boundary in the current frame) such that:

$$I(\mathbf{n}_i^t) = \begin{cases} \textit{inside}, & \textit{if } H_o(c) > H_b(c) + 0.1 \\ \textit{outside}, & \textit{if } H_b(c) > H_o(c) + 0.1 \\ \textit{undecided}, & \textit{else} \end{cases} \quad (4.20)$$

If we can not decide whether the node point \mathbf{n}_i^t is inside or outside of the actual object boundary, we then perform bi-directional search.

This selective search direction approach not only adds robustness to the algorithm but also reduces the search effort. The underlying assumption here is that the background colors are different from the object colors, at least around the object of

interest. An example of selective search direction is shown in Figure 4.6.

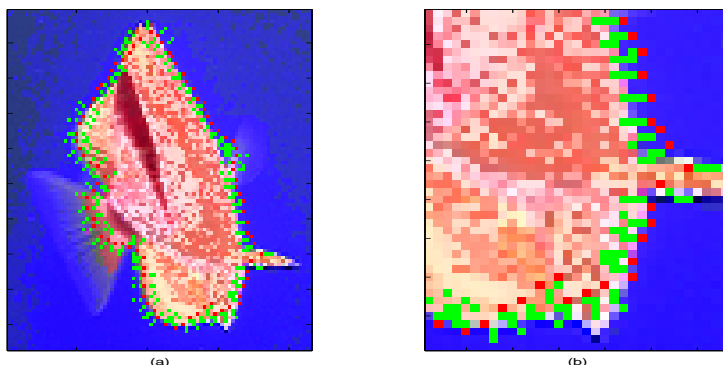


Figure 4.6. (a) The predicted boundary to be corrected is shown by red points. The selective search directions are shown by green lines (b) A close-up of (a)

4.4. Adaptation of Weights Using Performance Scores

In the literature there are no systematic ways to select the coefficients of the energy terms given in (4.13) and (4.14). However, these parameters are critical to the performance of the energy minimization. In this section, we present approaches to adjust the coefficients of the energy terms in a locally adaptive way using the performance evaluation measures given in Section 3.2.

The power of the proposed framework comes from the independent selection of the optimal weights for each sub-segment of the object contour. This optimal update strategy even includes complete disabling of some energy terms. In traditional snake formulations, a single weight value is assigned for each energy term for the entire object contour. Hence, an energy term, say motion energy, may help to improve the accuracy of a part of the contour, while distorting a different part of the contour, since reliability of the motion information varies across the contour. The proposed formulation enables us to emphasize different energy terms for different parts (segments) of the object contour in order to obtain the best possible results as indicated by quantitative performance measures. This section discusses the selection of the weight factors for each segment of the object contour.

4.4.1. Reliability Assessment of Color and Motion Boundaries

Before we try to snap the predicted object boundary to the color segmentation and motion segmentation boundaries, we should make sure that they are reliable.

Some parts the color segmentation boundaries may not coincide with actual object boundaries due to the inaccuracies of the color segmentation algorithm or because of the similarity of the object and background colors. We assess the reliability of the color segmentation boundaries using the color measure $d_{CB}(t; i)$ introduced in (3.16). First the color segmentation boundaries are divided into small segments and the color differences are found for each segment using (3.16). Then, the mean (M_C) and the standard deviation (S_C) of the $d_{CB}(t; i)$ values are found. The segments that have $d_{CB}(t; i)$ values below $M_C - k \cdot S_C$, where k is a positive number, are labeled as unreliable. The labeling to obtain a color segmentation reliability map is done by dilating the binary image consisting of the unreliable color segmentation boundary segments using a square structuring element. An example is shown in Figure 4.7. In Figure 4.7, we can

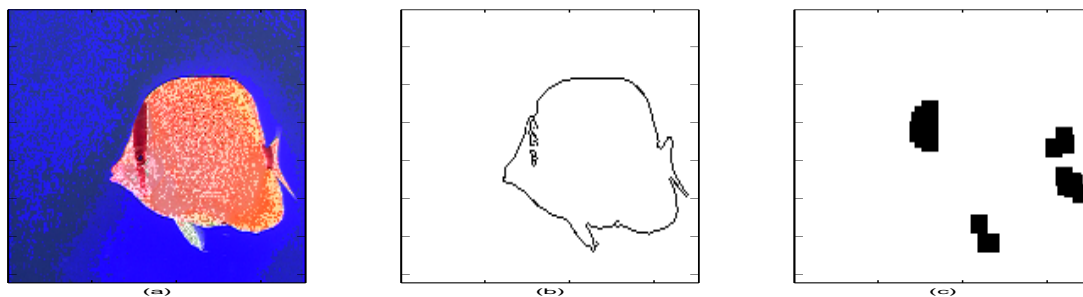


Figure 4.7. (a) Frame 127 of the “Bream” sequence (b) The color segmentation boundaries (c) The color segmentation reliability map in which dark regions indicate unreliable color segmentation boundaries

observe that, the unreliable segments are accumulated around several places, one of which is the dark red stripe above the eye of the fish. Since the color segmentation map incorrectly bends inward at the location where the red stripe and the blue background meet, the color distance at those locations is smaller compared to the other regions of the fish. Another region where unreliable segments are accumulated is the two sides of the rear fin since the fin is too thin to draw normal lines towards inside and outside of

the boundary.

Similarly, some parts of the motion segmentation boundaries may not be reliable due to the inaccuracies of one of the steps in the motion segmentation algorithm. We assess the reliability of the motion segmentation boundaries using the motion measure $d_M(t; i)$ introduced in (3.28). First the motion segmentation boundaries are divided into small segments and the motion differences are found for each segment using (3.28). Then, the mean (M_M) and the standard deviation (S_M) of the $d_M(t; i)$ values are found. The segments that have $d_M(t; i)$ values below $M_M - k \cdot S_M$, where k is a positive number, are labeled as unreliable.

4.4.2. Weight Adaptation

After the minimization of the overall energy (4.12), some segments of the estimated boundary of the object may not coincide with the actual color boundaries. This may indicate that the weight of the color energy term was not high enough. Similarly, if a segment of the boundary after energy minimization does not coincide with edges or motion boundaries we may conclude that the weight of the respective term was low.

Following this line of thought, the coefficients of the energy terms may be adjusted as follows:

$$\beta_{col,i} = g_1(d_{CB}(t; i)), \quad (4.21)$$

$$\beta_{mot,i} = g_2(d_M(t; i)) \quad (4.22)$$

where $d_{CB}(t; i)$ and $d_M(t; i)$ are defined in (3.16) and (3.28), respectively, for the estimated boundary. The functions $g_1(\cdot)$ and $g_2(\cdot)$ should be chosen such that, when $d_{CB}(t; i)$ or $d_M(t; i)$ are low at a certain segment, the corresponding weight should increase. For example, when the value of $d_{CB}(t; i)$ is low at a certain boundary segment, this indicates that that boundary segment does not coincide with a color boundary, that is, we are far from the color boundary. Then if we increase the weight of the color boundary energy term, we should be forcing the algorithm towards the nearest color

boundary faster.

We want to distribute the weight budget among the energy terms for each boundary segment i such that

$$\beta_{mot,i} + \beta_{col,i} + \beta_{edge,i} = 1. \quad (4.23)$$

Since motion segmentation boundaries are a subset of the edges and of the color segmentation boundaries, they are more likely to coincide with actual object boundaries if they exist and are reliable. Therefore, we give the top priority to the motion energy, i.e. if motion segmentation boundaries are nearby the current segment, we would like to go towards them first. Then, the selection strategy of β_{mot} is as follows:

$$\beta_{mot,i} = \frac{K_1}{d_M(t; i) + 1}, \quad (4.24)$$

where the parameter K_1 is set to zero if the distance of the current segment to the nearest motion segmentation boundary is above a certain value (e.g. 10 pixels) or if the current segment is in an unreliable motion segmentation boundary region. Otherwise, K_1 is set to 1. Considering that $d_M(t; i)$ takes values between 0 and 1, the values that $\beta_{mot,i}$ can take is:

$$\begin{cases} 0, & K_1 = 0 \\ 0.5 < \beta_{mot,i} < 1, & K_1 = 1 \end{cases} \quad (4.25)$$

Once we determine the motion weight in the above fashion, the weight of the color boundary energy term $\beta_{col,i}$ is determined as follows:

$$\beta_{col,i} = \frac{K_2(1 - \beta_{mot,i})}{d_{CB}(t; i) + 1}, \quad (4.26)$$

where the parameter K_2 is set to zero if the distance of the current segment to the nearest color segmentation boundary is above a certain value (e.g. 10 pixels) or if the current segment is in an unreliable color segmentation boundary region. Otherwise,

K_2 is set to 1. Considering that $d_{CB}(t; i)$ takes values between 0 and 1, the values that $\beta_{col,i}$ can take is:

$$\begin{cases} 0, & K_2 = 0 \\ \frac{1-\beta_{mot,i}}{2} < \beta_{col,i} < 1 - \beta_{mot,i}, & K_2 = 1 \end{cases} \quad (4.27)$$

Finally, the weight of the edge energy term is determined as follows:

$$\beta_{edge,i} = 1 - \beta_{mot,i} - \beta_{col,i} \quad (4.28)$$

The numerical range of the energy terms $E_{col,i}$, $E_{edge,i}$ and $E_{mot,i}$ are comparable since they are all calculated through the Chamfer distance transformation. If maximum search length is r for a certain node, then, the maximum value that each of the energy terms can take is approximately $4r$. The maximum value that the curvature energy term E_{curv} can take is 4 as given in (4.19). Therefore, the coefficient of the curvature term should be multiplied by the search range.

After the coefficients have been updated, the energy minimization is repeated forming the closed-loop. The algorithm exits from the loop when most of the node points become stationary between iterations or a maximum number of iterations is reached.

4.5. Experimental Results

The algorithm is tested on three video sequences containing rigid and non-rigid objects. The sequences use are “Bream”, “Coastguard” and “Parrot” sequences.

4.5.1. Results for the “Bream” Sequence

The Bream sequence contains a fish swimming and changing direction as shown in Figure 4.8. There is significant self-occlusion in this sequence as the fish turns from right to left. In the following experiments, the boundary is determined interactively

by the user in frame 100 and tracked automatically afterwards.

In the second row of Figure 4.8, the open-loop tracking results are given. That is, the boundary is updated based only on the boundary prediction scheme described in Section 4.2. Although the tracking results are good until frame 110, the predicted object boundary deviates significantly from the actual object boundary after frame 110.

In the third row of Figure 4.8, the boundary correction results are given using the edge and the curvature terms only with equal weighting, similar to traditional snake algorithms. The results are good until frame 115, but some deviations from the original boundary are observed as marked with square boxes, especially when the fish starts to turn from right to left.

In the fourth row of Figure 4.8, the closed-loop tracking results are shown with equal weighting of all the three energy term weights. Although the results are much better than the third row, the final boundary in frame 128 tends to snap to the incorrect color segmentation boundary as was discussed in Figure 4.7.

In the fifth row of Figure 4.8, the closed-loop tracking results using the proposed adaptive weighting strategy is shown. It can be observed that the outcome is definitely more satisfactory than the results shown in row 4. The problem at frame 128 is successfully eliminated.

In order to compare the four rows quantitatively, the number of misclassified pixels, the average color measure and the average distance from the actual object boundary for each frame are plotted in Figure 4.9. The average values of the plots for frames 101-129 are summarized in Table 4.1. The adaptive weighting strategy gives the least average number of misclassified pixels and the least average distance from the object boundary. The average color differences along the estimated object boundary is the highest for the adaptive weight method, indicating a good fit to color boundaries.

4.5.2. Results for the “Parrot” Sequence

Another sequence that is used to test the algorithm is the Parrot sequence, the first and the 18th frames of which are shown in the first row of Figure 4.10. The amount of motion from frame 1 to frame 18 is approximately (26,-20) pixels in the x and y directions. The object and the background in this sequence is quite cluttered.

The second row of Figure 4.10 shows the closed-loop tracking results using only edge and curvature terms. The third row shows the closed-loop tracking results using all the energy terms with equal weighting. Finally, the fourth row shows the closed-loop tracking results using all the energy terms with adaptive weighting. If we compare the equal weight and adaptive weight results, especially the sections marked with boxes, we can see that the adaptive weighting strategy is more successful in tracking the left wing of the Parrot. The equal weighting strategy is affected by the imperfection of the color-refined motion segmentation boundaries which are shown in Figure 4.11(c). However, the adaptive weighting strategy overcomes this problem by making the weight of the motion energy term zero in those regions as shown in Figure 4.11(d). In this figure, the boundary sections drawn with green lines indicate the regions where the color-refines motion segmentation boundaries are found unreliable and edges are given more weight. The boundary sections drawn with white lines indicate the regions where the color-refined motion segmentation boundaries are reliable and more weight is given to the motion energy term.

In order to compare the results quantitatively, the number of misclassified pixels, the average color measure and the average distance from the actual object boundary for each frame are plotted in Figure 4.12. A quantitative comparison table is given also in Table 4.2. It can be observed from the table that the number of misclassified pixels and the average distance from the actual object boundary is the smallest for the proposed adaptive energy weighting method.

4.5.3. Results for the “Coastguard” Sequence

Another sequence used in the experiments is the CoastGuard sequence which is shown in Figure 4.13. The boat object in this sequence can be considered to be rigid. The background is quite cluttered in this sequence. There is also severe blurring when the camera moves suddenly around frame 72 as can be seen in Figure 4.13.

The open-loop tracking results with the imposed rigidity constraint is given in Figure 4.14. Feature points which have been successfully tracked at each frame are shown with yellow boxes and the predicted feature points are shown with red boxes. Blue lines denote the motion vectors of the tracked feature points. The feature points inside the segmentation map of the boat are used to estimate the motion of the boundary of the object. A filtering of the motion vectors is helpful if there are outlier motion vectors. The algorithm tracks the boat successfully despite the heavy blurring due to camera movement in frame 72.

For sequences with rigid objects, the boundary prediction block without the boundary correction step can be sufficient as demonstrated in this example.

Table 4.1. Summary of quantitative comparisons for the “Bream” sequence

	Average number of misclassified pixels	Average color measure	Average pixel distance from actual object boundary
Open Loop	1468	0.206	5.02
Edges Only	298	0.317	1.55
Equal Weight	223	0.327	1.24
Adaptive Weight	218	0.330	1.17

Table 4.2. Summary of quantitative comparisons for the “Parrot” sequence

	Average number of misclassified pixels	Average color measure	Average pixel distance from actual object boundary
Edges Only	1006	0.180	3.29
Equal Weight	905	0.184	2.72
Adaptive Weight	804	0.182	2.38

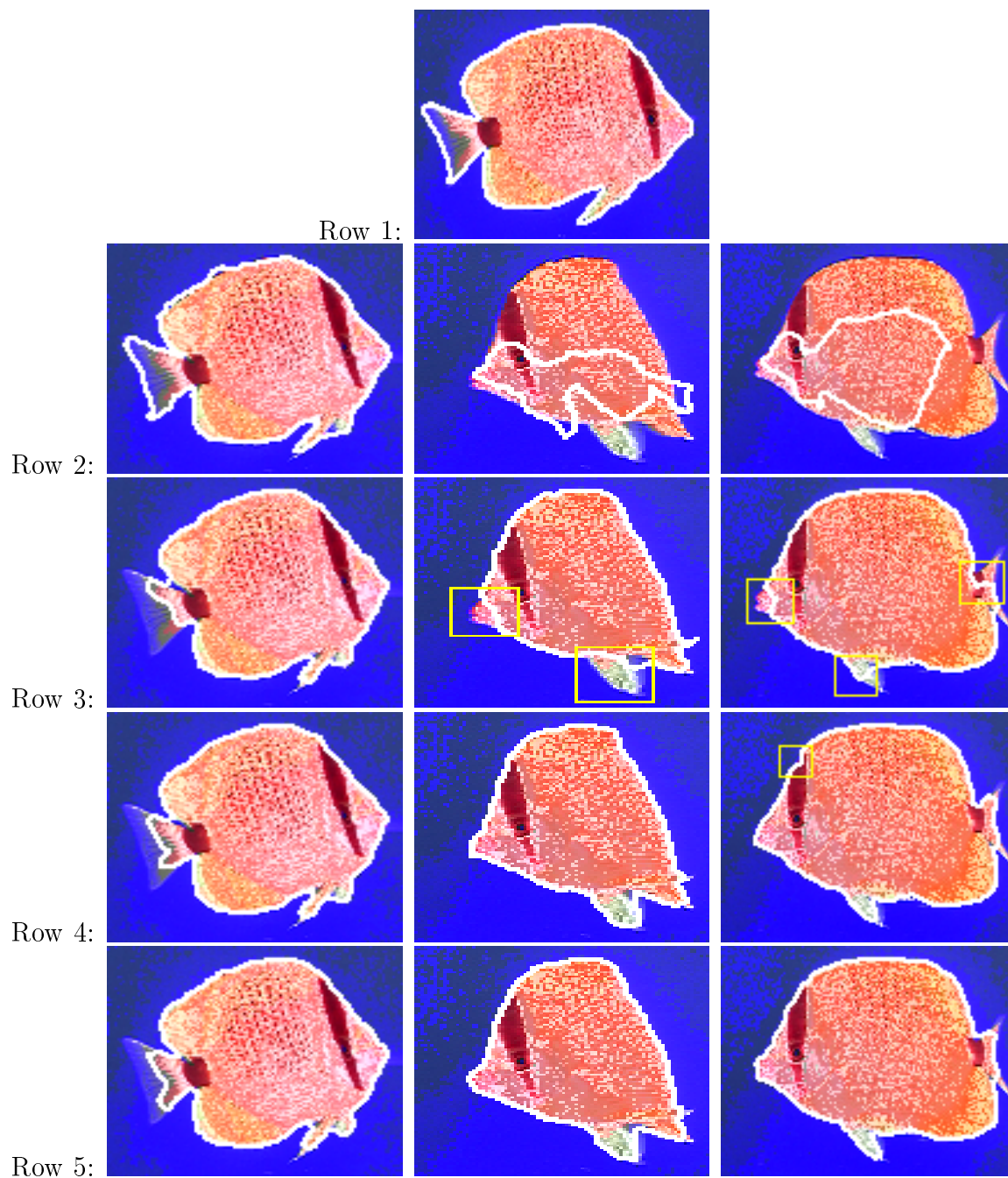
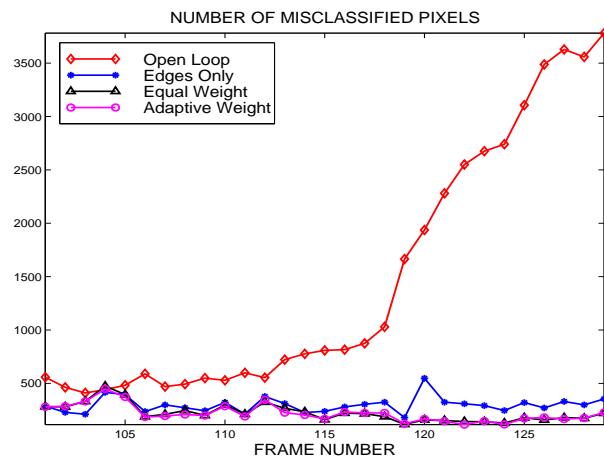
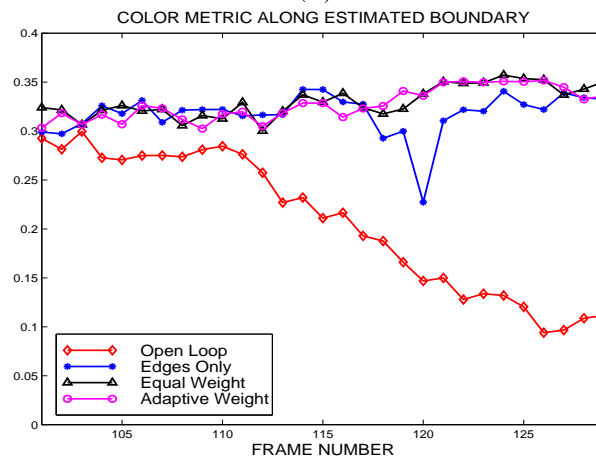


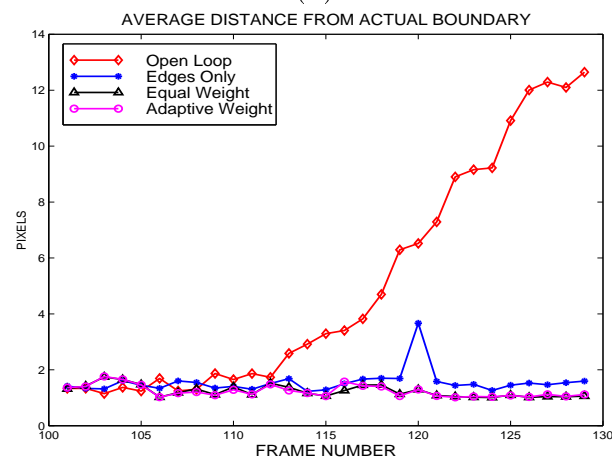
Figure 4.8. Row 1: Hand-drawn boundary at frame 100. Row 2 - Row 5: Tracking results using open-loop, and closed loop with edge energy only, equal weighting and adaptive weighting methods, respectively, for frames 109, 121, and 128



(a)



(b)



(c)

Figure 4.9. (a) The number of misclassified pixels for each frame of 'Bream' (b) The color measure along the estimated boundary for each frame (c) The average distance from actual object boundary for each frame

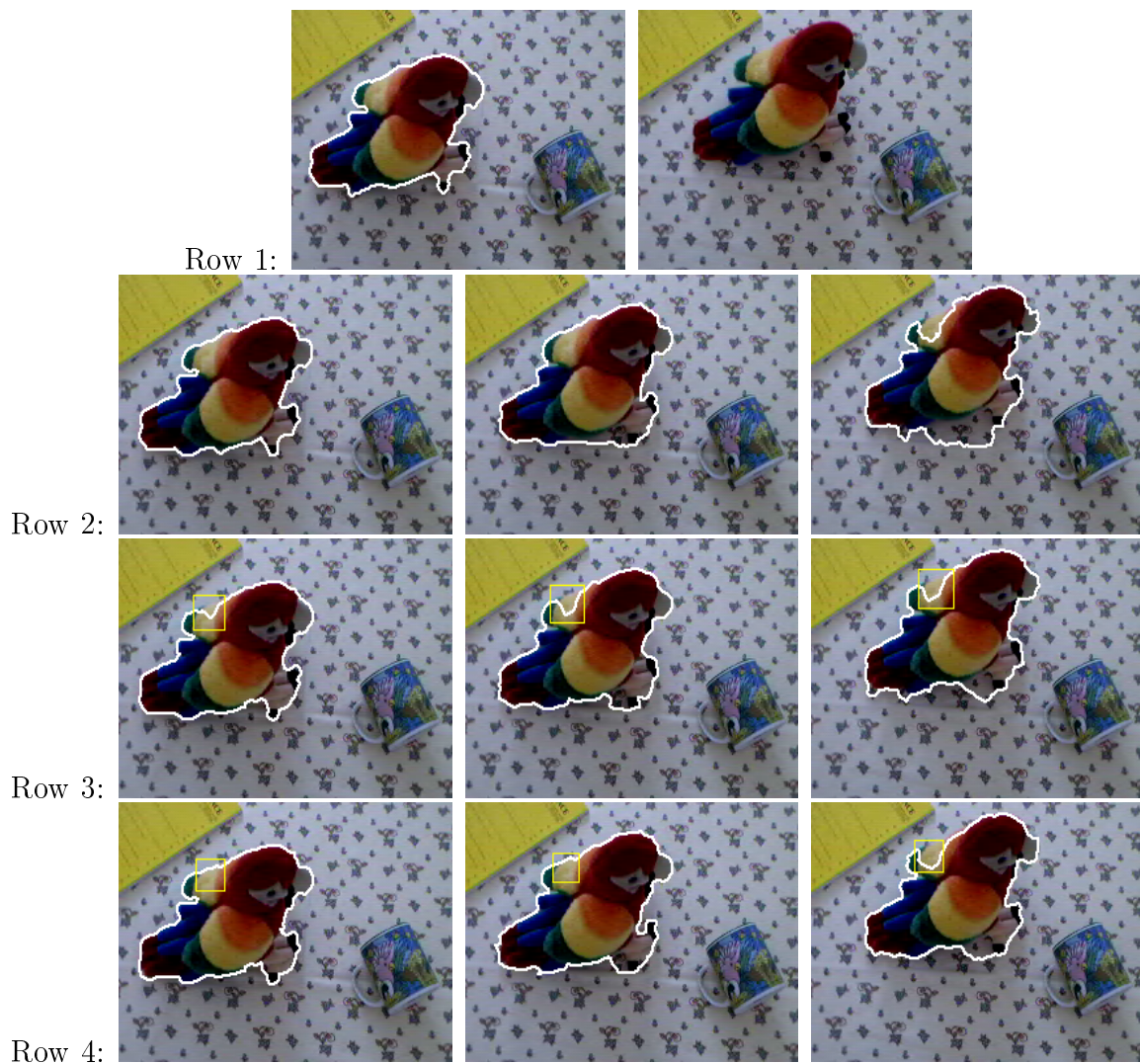


Figure 4.10. Row 1: Hand-drawn boundary at frame 1. Row 2 - Row 4: Tracking results using closed-loop with edge energy only, equal weighting and adaptive weighting methods, respectively, for frames 2, 8, and 18

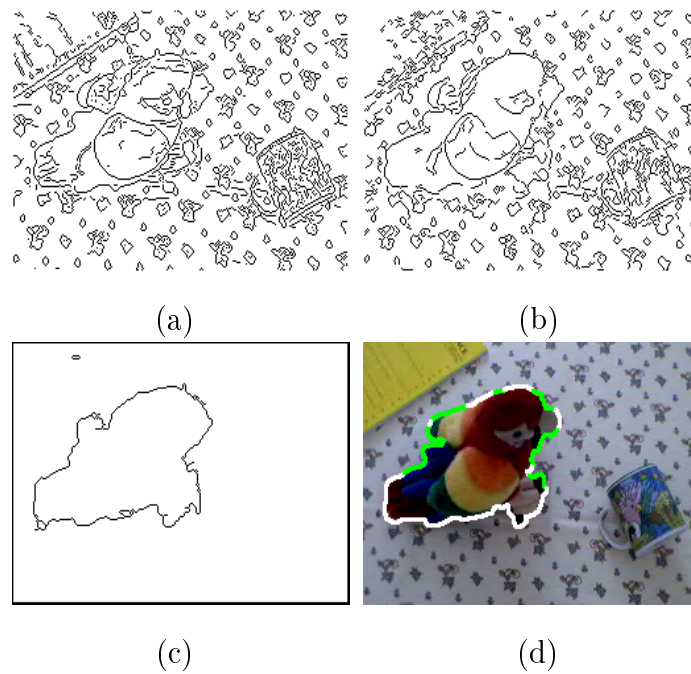
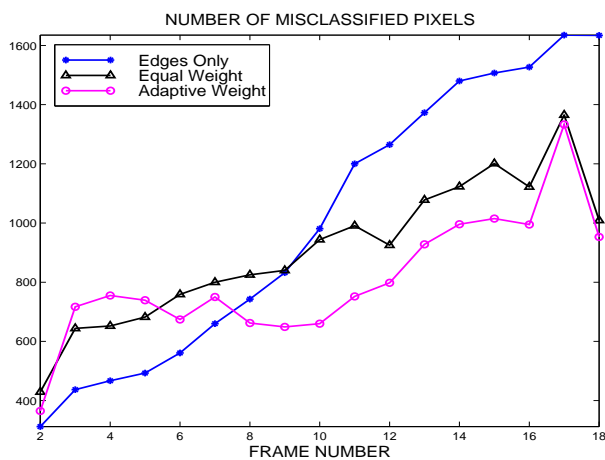
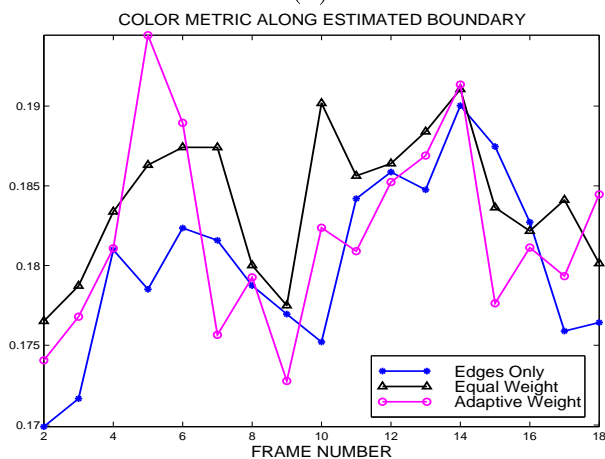


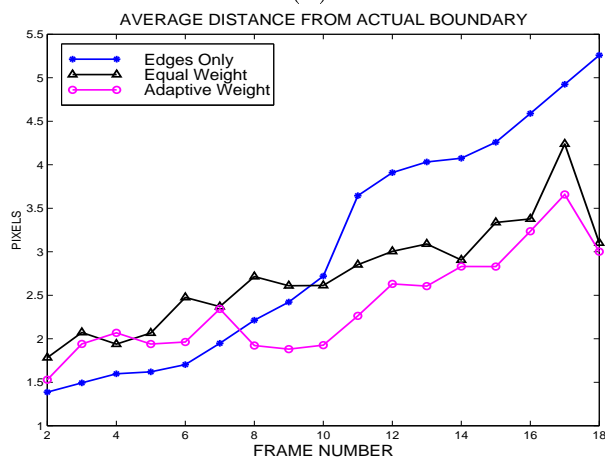
Figure 4.11. (a) Edges of the second frame (b) Color segmentation boundaries (c) Color-refined motion segmentation boundaries (d) Green segments: zero motion energy weight. White segments: weight of the motion energy term is the largest



(a)



(b)



(c)

Figure 4.12. (a) The number of misclassified pixels for “Parrot” sequence (b) Color difference measure along boundary (c) Average distance from the actual object boundary for each frame



Figure 4.13. Frames 1, 45, 72, and 86 of the “Coast Guard” sequence

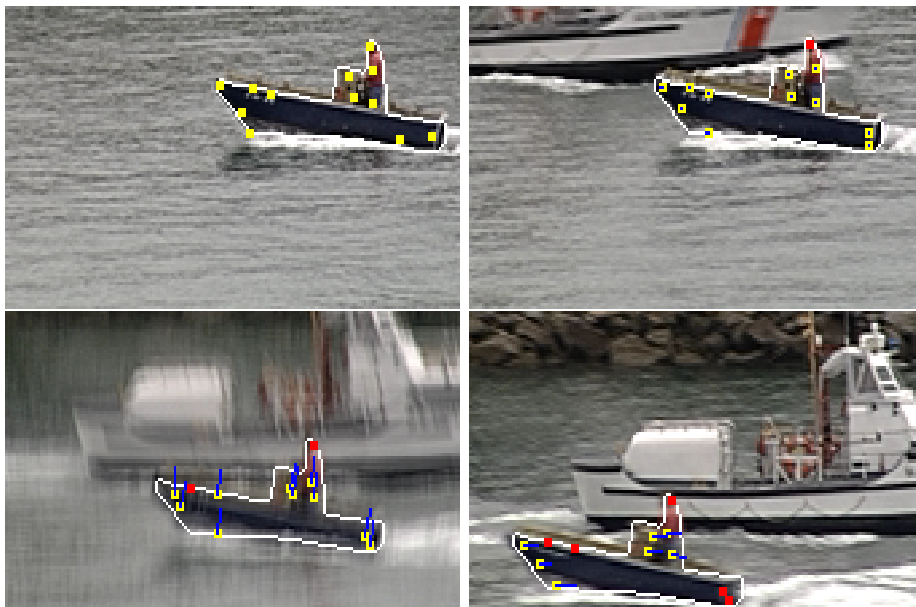


Figure 4.14. Open-loop tracking results for frames 1, 45, 72, and 86. Tracked and predicted feature points are shown by yellow and red boxes, respectively. Motion vectors are shown by blue lines

5. CONCLUSIONS AND FUTURE DIRECTIONS

5.1. Conclusions

In this thesis, a survey of existing segmentation and tracking methods is given. Video object segmentation and tracking is a very important problem due to its applications in many fields. The existing literature is analyzed in two groups, namely, fully automatic methods and semi-automatic methods. Fully automatic object segmentation and tracking are very challenging tasks, and generally require strict assumptions such as stationarity of the background or known shape space for the objects. In semi-automatic methods, the type of user interaction can be in the form of contour-based interaction or region-based interaction.

Although there are many methods in the literature for video object segmentation and tracking, there are only a few methods for evaluating their performances quantitatively. We introduce novel performance evaluation measures for video object segmentation and tracking both when the ground truth (reference) segmentation maps are available and when they are not available. Three performance evaluation measures are introduced that use ground-truth segmentation measures. They rely on pixel misclassification penalty, shape penalty and motion penalty. These three measures can also be combined to give a single quantitative measure for the video frame, video shot or the whole video sequence. They can furthermore be used for temporal and spatial localization of incorrectly segmented regions. The performance measures are tested on real and synthetic image sequences and are shown to agree with the subjective quality assessments of the three leading segmentation algorithms found in the literature.

Two types of performance evaluation measures are introduced that do not use ground-truth segmentation maps, namely, the color measures and the motion measure. The color measures are further divided into two groups as the color difference measure along the estimated object boundary and the inter-frame video object histogram difference measures. Experiments show that the proposed measures are indeed respon-

sive to errors in the correct segmentation map. Evaluation of the same set of object tracking results using measures with and without ground-truth segmentation maps show that the performance measures without ground-truth data are highly correlated with the performance measures using ground-truth data. The set of linear transformations which maximize this correlation are also determined using canonical correlation analysis.

A scalable, semi-automatic video object tracking method which utilizes the proposed performance evaluation measures is also introduced. The method consists of two main processing stages. The first stage is the *boundary prediction* step, which tracks the coarse boundary of the object using a fast feature point tracking approach. The second stage is the *boundary correction* step, which gives the pixel-accurate object boundary by using an energy minimization scheme. A feedback loop is utilized to adjust the weights of each energy term locally for each boundary segment using the developed performance evaluation measures. Energy terms which are derived from color segmentation and color-refined motion segmentation boundaries are introduced, which are inherently normalized because of their definition using Chamfer distance transform. Experimental results demonstrate the superiority of the proposed adaptive weight algorithm to previous fixed-weight approaches.

We also introduce novel algorithms for image segmentation and motion estimation. We present a segmentation algorithm in the presence of local illumination variations using fuzzy c-planes clustering. The mean intensity value used in the classical fuzzy c-means algorithm is replaced with a plane fitting approach to take into account the spatial illumination variations. The method may be extended to the more general “fuzzy c-surfaces” scheme by fitting higher order surfaces instead of planes. The method includes a local regularization scheme for the updating of the membership functions. The main idea behind the local regularization is to make the membership of an image pixel closer to the mean of the memberships of its neighbors, but avoiding smoothing over large differences between the neighbors, which may be due to genuine segment boundaries. The experimental results on gray-scale and color images demonstrate segmentation improvement in regions where there are local illumination

variations.

Another video processing task for which we utilize the fuzzy *c*-planes clustering method is a frequency domain motion estimation algorithm. We present a novel approach for the pairing and motion parameter estimation problem when the motion estimation problem is handled in a harmonic retrieval framework. The new approach is based on fitting planes to the 3-D data in the frequency domain using the fuzzy *c*-planes algorithm. The parameters of these planes correspond to the horizontal and vertical motion parameters of the objects in the scene. This approach eliminates the need to pair the velocity components of the objects. The experiments on synthetic and natural image sequences demonstrate the effectiveness of the proposed method.

5.2. Future Directions

Possible future research directions to continue the work of this thesis are:

- Develop a robust method for automatic initialization of an object boundary in the first frame of a video sequence.
- Incorporate a priori shape space information where possible to improve the accuracy and robustness of the tracking algorithm.
- Extend the tracking algorithm to track multiple objects, especially if their histograms are not different and when they occlude each other.
- Extend the performance evaluation measures so that they are invariant to illumination variations in a video sequence.
- Optimize the software implementation of the proposed tracking method to enable real-time tracking.

REFERENCES

1. Chiariglione, L., “The MPEG-4 Standard”, *Journal of China Institute of Communications*, pp. 54–67, 1998.
2. Koenen, R., F. Pereira and L. Chiariglione, “MPEG-4: Context and Objectives”, *Signal Processing: Image Communication*, Vol. 9, pp. 295–304, 1997.
3. Koenen, R., *MPEG-7 Context and Objectives*, Tech. rep., ISO/IEC/SC29/WG11 N2326, Dublin, July 1998.
4. Sikora, T., “The MPEG-4 Video Standard Verification Model”, *IEEE Transactions Circuits and Systems for Video Technology*, pp. 82–100, 1997.
5. Tekalp, M., P. V. Beek, C. Toklu and B. Gunsel, “Two-Dimensional Mesh-Based Visual-Object Based Representation for Interactive Synthetic/Natural Digital Video”, *Proceedings of the IEEE*, Vol. 86, No. 6, pp. 1029–1051, 1998.
6. Irani, M. and S. Peleg, “Motion Analysis for Image Enhancement: Resolution, Occlusion, and Transparency”, *Journal of Visual Communication and Image Representation*, Vol. 4, No. 4, pp. 324–335, 1993.
7. Wang, J. Y. A. and E. H. Adelson, “Representing Moving Images with Layers”, *IEEE Trans. Image Processing*, Vol. 3, No. 5, pp. 625–638, 1994.
8. Borshukov, G. D., G. Bozdağı, Y. Altunbaşak and A. M. Tekalp, “Motion Segmentation by Multistage Affine Classification”, *IEEE Trans. Image Processing*, Vol. 6, No. 11, pp. 1591–1594, 1997.
9. Chang, M. M., A. M. Tekalp and M. İbrahim Sezan, “Simultaneous Motion Estimation and Segmentation”, *IEEE Trans. Image Processing*, Vol. 6, No. 9, pp. 1326–1333, 1997.

10. Cham, T. and J. M. Regh, "A Multiple Hypothesis Approach to Figure Tracking", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 239–245, 1999.
11. Ju, S. X., M. J. Black and Y. Yacoob, "Cardboard People: A Parametrized Model of Articulated Image Motion", *Proc. Int. Conf. Face and Gesture Recognition*, pp. 561–567, 1996.
12. Bregler, C. and J. Malik, "Tracking People with Twists and Exponential Maps", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 239–245, Santa Barbara, CA, June 1998.
13. Wachter, S. and H. H. Nagel, "Tracking Persons in Monocular Image Sequences", *Computer Vision and Image Understanding*, Vol. 74, No. 3, pp. 174–192, June 1999.
14. Blake, A. and M. Isard, *Active Contours*, Springer-Verlag, 1998.
15. Baumberg, A. M., *Learning Deformable Models for Tracking Human Motion*, Ph.D. Thesis, The University of Leeds, School of Computer Studies, October 1995.
16. Isard, M. and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking", *International Journal of Computer Vision*, Vol. 29, No. 1, pp. 5–28, 1998.
17. Yacoob, Y. and L. S. Davis, "Learned Models for Estimation of Rigid and Articulated Human Motion from Stationary or Moving Camera", *International Journal of Computer Vision*, Vol. 12, No. 1, pp. 5–30, 2000.
18. Wren, C. R., A. Azerbayejani, T. Darrell and A. P. Pentland, "Pfinder: Real-time Tracking of the Human Body", *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 19, No. 7, pp. 780–785, July 1997.

19. Haritaoglu, I., D. Harwood and L. S. Davis, "W-4: Real-time Surveillance of People and Their Activities", *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 22, No. 8, pp. 809–830, 2000.
20. Moscheni, F., S. Bhattacharjee and M. Kunt, "Spatiotemporal Segmentation Based on Region Merging", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 20, No. 9, pp. 897–915, 1998.
21. Hotter, M. and R. Thoma, "Image Segmentation Based on Object Oriented Mapping Parameter Estimation", *Signal Processing*, Vol. 15, pp. 315–334, 1988.
22. Ma, S., S. Chen and L. Chen, "An Efficient Moving Object Segmentation Algorithm for MPEG-4 Encoding Systems", *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (ICASSP)*, 2000.
23. Liu, T. and F. Qi, "A New Technique for Image Segmentation: Region Binding", *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (ICASSP)*, 2000.
24. Vincent, L. and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations", *IEEE Trans. Pattern Analysis, and Machine Intelligence*, Vol. 13, pp. 583–598, 1991.
25. Neri, A., S. Colonnese, G. Russo and P. Talone, "Automatic Moving Object and Background Separation", *Signal Processing*, Vol. 66, No. 2, pp. 219–232, 1998.
26. Shi, J. and C. Tomasi, "Good Features to Track", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.
27. Meier, T. and K. N. Ngan, "Automatic Segmentation of Moving Objects for Video Object Plane Generation", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol. 8, No. 5, pp. 525–538, 1998.
28. Altunbaşak, Y., P. E. Eren and A. M. Tekalp, "Region-Based Parametric Motion

- Segmentation Using Color Information”, *Graphical Models and Image Processing*, Vol. 60, No. 1, pp. 13–23, 1998.
29. Castagno, R., T. Ebrahimi and M. Kunt, “Video Segmentation Based on Multiple Features for Interactive Multimedia Applications”, *IEEE Trans. On Circuits and Systems for Video Technology*, Vol. 8, No. 5, pp. 562–571, 1998.
 30. Bezdek, J. C. and N. R. Pal, “Some New Indexes of Cluster Validity”, *IEEE Trans. on Systems, Man and Cybernetics – Part B: Cybernetics*, Vol. 28, No. 3, pp. 301–315, 1998.
 31. Gath, I. and B. Geva, “Unsupervised Optimal Fuzzy Clustering”, *IEEE Trans. Pattern Analysis, and Machine Intelligence*, Vol. 11, No. 7, pp. 773–781, 1989.
 32. Otsu, N., “A Threshold Selection Method From Gray Level Histograms”, *IEEE Trans. on Systems, Man, and Cybernetics (SMC)*, Vol. 9, No. 1, pp. 62–66, 1979.
 33. Comaniciu, D., V. Ramesh and P. Meer, “Real-Time Tracking of Non-Rigid Objects Using Mean Shift”, *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.
 34. Toklu, C., *Object-based Digital Video Processing Using 2-D Meshes*, Ph.D. Thesis, University of Rochester, Rochester, New York, 1998.
 35. Eren, P. E. and A. M. Tekalp, “Keyframe-Based Bi-directional 2-D mesh Representation for Video Object Tracking and Manipulation”, *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 1999.
 36. Toklu, C., A. T. Erdem, M. I. Sezan and A. M. Tekalp, “Tracking Motion and Intensity Variations Using Hierarchical 2-D Mesh Modeling”, *Graphical Models and Image Processing*, Vol. 58, No. 6, pp. 553–573, 1996.
 37. Gu, C. and M. Lee, “Semantic Video Object Segmentation and Tracking using Mathematical Morphology and Perspective Motion Model”, *Proc. IEEE Int.*

- Conf. on Image Processing (ICIP)*, Vol. II, pp. 514–517, 1997.
38. Fu, Y., A. T. Erdem and A. M. Tekalp, “Occlusion Adaptive Motion Snake”, *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 1998.
 39. Luo, H. and A. Eleftheriadis, “Spatial Temporal Active contour Interpolation for Semi-automatic Video Object Generation”, *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 1999.
 40. Fu, Y., A. T. Erdem and A. M. Tekalp, “Tracking Visible Boundary of Objects Using Occlusion Adaptive Motion Snake”, *IEEE Transactions on Image Processing*, Vol. 9, No. 12, pp. 2051–2060, December, 2000.
 41. Nascimento, J. C., A. J. Abrantes and J. S. Marques, “An Algorithm for Centroid-based Tracking of Moving Objects”, *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (ICASSP)*, 1999.
 42. Zhong, D. and S. F. Chang, “Video Object Model and Segmentation for Content-based Video Indexing”, *Proceedings of IEEE International Conference on Circuits and Systems*, 1997.
 43. Zanoguera, F., B. Marcotegui and F. Meyer, “A Toolbox for Interactive Segmentation Based on Nested Partitions”, *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 1999.
 44. Marcotegui, B., F. Zanoguera, P. Correia, R. Rosa, F. Marques, R. Mech and M. Wollborn, “Video Object Generation Tool Allowing Friendly User Interaction”, *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 1999.
 45. Mansouri, A. and J. Konrad, “Minimum Description Length Region Tracking with Level Sets”, *IST/SPIE Symposium on Electronic Imaging, Image and Video Comm.*, pp. 23–28, San Jose, CA, January 2000.
 46. Mansouri, A., A. Olivier and J. Konrad, “Minimum Description Length Region

- Tracking with Level Sets”, *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2000.
47. Pal, N. R. and S. K. Pal, “A Review on Image Segmentation Techniques”, *Pattern Recognition*, Vol. 26, No. 9, pp. 1277–1294, 1993.
 48. Fu, K. S. and J. K. Mui, “A Survey on Image Segmentation”, *Pattern Recognition*, Vol. 13, pp. 3–6, 1981.
 49. Haralick, R. M. and L. G. Shapiro, “Survey: Image Segmentation Techniques”, *Computer Vision, Graphics, and Image Processing*, Vol. 24, pp. 100–132, 1985.
 50. Sahoo, P. K., S. Soltani, A. K. C. Wong and Y. C. Chen, “A Survey of Thresholding Techniques”, *Computer Vision, Graphics, and Image Processing*, Vol. 41, pp. 233–260, 1988.
 51. Skarbek, W. and A. Koschan, “Color Image Segmentation: A Survey”, *Technical Report 94-32, Technische Universitat Berlin*, 1994.
 52. Kurugöllü, F., *Color Image Segmentation*, Ph.D. thesis, İstanbul Technical University, 2000.
 53. Gonzales, R. C. and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Massachusetts, 1992.
 54. Toliaş, Y. A. and S. M. Panas, “On Applying Spatial Constraints in Fuzzy Image Clustering Using a Fuzzy Rule-based System”, *IEEE Signal Processing Letters*, Vol. 5, No. 10, pp. 245–247, 1998.
 55. Huntsberger, T. L., C. L. Jacobs and L. Cannon, “Iterative Fuzzy Image Segmentation”, *Pattern Recognition Letters*, Vol. 18, No. 2, pp. 131–138, 1985.
 56. Pham, D. L. and J. L. Prince, “An Adaptive Fuzzy C-means Algorithm for Image Segmentation in the Presence of Intensity Inhomogeneities”, *Pattern Recognition*

- Letters*, Vol. 20, pp. 57–68, 1999.
57. Pappas, T. N., “An Adaptive Clustering Algorithm for Image Segmentation”, *IEEE Transactions on Signal Processing*, Vol. 40, No. 4, pp. 901–914, 1992.
 58. Salembier, P., “Morphological Multiscale Segmentation for Image Coding”, *Signal Processing*, Vol. 38, pp. 359–386, 1994.
 59. Salembier, P., P. Brigger, J. R. Casas and M. Pardas, “Morphological Operators for Image and Video Compression”, *IEEE Transactions on Image Processing*, Vol. 5, No. 6, 1996.
 60. Lim, D. K. and Y. S. Lo, “Image Segmentation Using Hierarchical Meshes”, *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 1999.
 61. Liu, J. and Y. Yang, “Multiresolution Color Image Segmentation”, *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 16, No. 7, 1994.
 62. Luo, J., R. T. Gray and H. Lee, “Incorporation of Derivative Priors in Adaptive Bayesian Color Image Segmentation”, , 1999.
 63. Bhanu, B., L. Sungkee and J. Ming, “Adaptive Image Segmentation Using a Genetic Algorithm”, *IEEE Trans. on Systems, Man, and Cybernetics (SMC)*, Vol. 25, No. 12, pp. 1543–1567, 1995.
 64. Haseyama, M., M. Kumagai and H. Kitajima, “A Genetic Algorithm Based Image Segmentation”, *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (ICASSP)*, 1999.
 65. Andrey, P. and P. Tarraux, “Unsupervised Image Segmentation Using a Distributed Genetic Algorithm”, *Pattern Recognition*, Vol. 27, No. 5, pp. 659–673, 1994.
 66. Li, J., A. Najmi and R. M. Gray, “Image Classification by a Two Dimensional Hid-

- den Markov Model”, *Proceedings of IEEE International Conference on Acoustics Speech Signal Processing (ICASSP)*, 1999.
67. Mackeown, W. P. J., P. Greenway, B. T. Thomas and W. A. Wright, “Contextual Image Labelling with a Neural Network”, *IEE Proc.-Vis.Image Signal Process.*, Vol. 141, No. 4, pp. 238–244, 1994.
68. Chu, C. and J. K. Aggarwal, “The Integration of Image Segmentation Maps Using Region and Edge Information”, *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 15, No. 12, pp. 1241–1252, 1993.
69. Saber, E., A. M. Tekalp and G. Bozdağı , “Fusion of Color and Edge Information for Improved Segmentation and Edge Linking”, *Image and Vision Computing*, Vol. 15, pp. 769–780, 1997.
70. Chehikian, A., “Image Segmentation by Contours and Regions Cooperation”, *Signal Processing*, Vol. 78, pp. 329–347, 1999.
71. Haddon, J. F. and J. F. Boyce, “Image Segmentation by Unifying Region and Boundary Information”, *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 12, No. 10, pp. 929–948, 1990.
72. Gershon, R. and A. D. Jepson, “Ambient Illumination and the Determination of Material Changes”, *Journal of the Optical Society of America A*, Vol. 3, No. 10, pp. 1700–1707, 1986.
73. Tominaga, S. and B. A. Wandell, “Standard Surface-Reflectance Model and Illuminant Estimation”, *Journal of the Optical Society of America A*, Vol. 6, No. 4, pp. 576–584, 1989.
74. Tominaga, S. and B. A. Wandell, “Component Estimation of Surface Spectral Reflectance”, *Journal of the Optical Society of America A*, Vol. 7, No. 2, 1990.
75. Borsotti, M., P. Campdelli and P. Schettini, “Quantitative Evaluation of Color

- Image Segmentation Results”, *Pattern Recognition Letters*, Vol. 19, pp. 741–747, 1998.
76. Zhang, Y. J., “Evaluation and Comparison of Different Segmentation Algorithms”, *Pattern Recognition Letters*, Vol. 18, pp. 963–974, 1997.
77. Levine, M. D. and A. M. Nazif, “Dynamic Measurement of Computer Generated Image Segmentations”, *IEEE Trans. On Pattern Recognition and Machine Intelligence*, Vol. 7, No. 2, pp. 155–164, 1985.
78. Goehrig, G. and L. Ledford, “Analysis of Image Segmentation Approaches with Emphasis on Performance Evaluation Criteria”, *Proceedings of SPIE*, Vol. 252, pp. 124–129, 1980.
79. Rees, G., P. Greenway and D. Morray, “Metrics for Image Segmentation”, *SPIE Conference On Visual Information Processing VII*, Vol. 3387, 1998.
80. Yasnoff, W. A., J. K. Mui and J. W. Bacus, “Error Measures for Scene Segmentation”, *Pattern Recognition*, Vol. 9, pp. 217–231, 1977.
81. Zhang, Y. J., “A Survey on Evaluation Methods for Image Segmentation”, *Pattern Recognition*, Vol. 29, No. 8, pp. 1335–1346, 1996.
82. Weszka, J. S. and A. Rosenfeld, “Threshold Evaluation Techniques”, *IEEE Transactions on Systems, Man, and Cybernetics (SMC)*, Vol. 8, No. 8, pp. 622–629, 1978.
83. Hoppner, F., F. Klawonn, R. Kruse and T. Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*, John Wiley and Sons, 1999.
84. Dawant, B. M., A. P. Zijdenbos and R. A. Margolin, “Correction of Intensity Variations in MR Images for Computer-aided Tissue Classification”, *IEEE Trans. on Medical Imaging*, Vol. 12, pp. 770–781, 1993.

85. Johnson, B., M. S. Atkins, B. Mackiewich and M. Anderson, "Segmentation of Multiple Sclerosis Lesions in Intensity Corrected Multispectral MRI", *IEEE Trans. on Medical Imaging*, Vol. 15, pp. 154–169, 1996.
86. Meyer, C. R., H. B. Peyton and J. Pipe, "Retrospective Correction of Intensity Inhomogeneities", *IEEE Trans. on Medical Imaging*, Vol. 14, pp. 36–41, 1995.
87. Unser, M., "Multigrid Adaptive Image Processing", *Proc. IEEE Int. Conf. on Cybernetics Society*, pp. 163–165, 1975.
88. Erdem, C. E. and B. Sankur, "Illumination Invariant Fuzzy Image Segmentation", *Proc. IEEE Balkan Conf. Signal Processing, Communications, Circuits and Systems*, Turkey, 2000.
89. Bezdek, J. C., "A Convergence Theorem for the Fuzzy ISODATA Clustering", *IEEE Trans. Pattern Analysis, and Machine Intelligence*, Vol. 2, pp. 1–8, 1980.
90. Dunn, J. C., "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-separated Clusters", *J. Cybernetics*, Vol. 3, pp. 32–57, 1973.
91. Windham, M. P., "Cluster Validity for the Fuzzy C-means Clustering Algorithm", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 4, No. 4, pp. 357–363, 1982.
92. Cosic, D. and S. Loncaric, "New Methods for Cluster Selection in Unsupervised Fuzzy Clustering", *Automatika*, Vol. 37, pp. 133–137, 1990.
93. Chen, W., G. B. Giannakis and N. Nandhakumar, "Spatio-Temporal Approach for Time-Varying Image Motion Estimation", *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Vol. 2, pp. 232–236, 1994.
94. Barron, J. L., D. J. Fleet and S. S. Beauchemin, "Performance of Optical Flow Techniques", *International Journal of Computer Vision*, Vol. 12, No. 1, pp. 43–77, 1994.

95. Horn, B. K. P. and B. G. Schunk, "Determining Optical Flow", *Artificial Intelligence*, Vol. 17, pp. 185–203, 1981.
96. Tekalp, A. M., *Digital Video Processing*, Prentice-Hall, New Jersey, 1995.
97. Young, R. W. and N. G. Kingsbury, "Frequency-Domain Motion Estimation Using a Complex Lapped Transform", *IEEE Transactions on Image Processing*, Vol. 2, No. 1, pp. 2–17, 1993.
98. Demiroğlu, C. and E. Anarım, "Sıklık Düzleminde Devinim Parametrelerinin Keşirimi", *Sinyal İşleme ve Uygulamaları Kurultayı*, Ankara, pp. 336–339, 1999.
99. Chen, W., G. Giannakis and N. Nandhakumar, "A Harmonic Retrieval Framework for Discontinuous Motion Estimation", *IEEE Transactions on Image Processing*, Vol. 7, No. 9, pp. 1242–1256, 1998.
100. Erdem, C. E., G. Z. Karabulut and E. A. E. Yanmaz, "Motion Estimation in the Frequency Domain Using Fuzzy C-Planes Clustering", *IEEE Transactions on Image Processing*, Vol. 10, No. 12, pp. 1873–1879, December 2001.
101. Bezdek, J. C., C. Coray, R. Gunderson and J. Watson, "Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy C-lines", *SIAM Journal of Appl. Math.*, Vol. 40, No. 2, pp. 339–357, 1981.
102. Bezdek, J. C., C. Coray, R. Gunderson and J. Watson, "Detection and Characterization of Cluster Substructure II. Fuzzy C-varieties and Convex Combinations Thereof", *SIAM Journal of Appl. Math.*, Vol. 40, No. 2, pp. 358–372, 1981.
103. Stoica, P. and R. Moses, *Introduction to Spectral Analysis*, Prentice Hall, 1997.
104. Haykin, S., *Adaptive Filter Theory*, Prentice Hall, 1991.
105. Senior, A., A. Hampapur, Y. Tian, L. Brown, S. Pankanti and R. Bolle, "Appearance Models for Occlusion Handling", *Proc. Second IEEE Workshop on Perfor-*

- mance Evaluation of Tracking and Surveillance*, 2000.
106. Erdem, C. E. and B. Sankur, "Performance Evaluation Metrics for Object-Based Video Segmentation", *Proc. X European Signal Processing Conference*, Vol. 2, pp. 917–920, September 2000.
 107. Strasters, K. C., "Three-dimensional Image Segmentation Using a Split, Merge and Group Approach", *Pattern Recognition Letters*, Vol. 12, pp. 307–325, 1991.
 108. Arkin, E. M., L. P. Chew, D. P. Huttenlocker, K. Kedem and J. S. B. Mitchell, "An Efficient Computable Metric for Comparing Polygonal Shapes", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 209–215, 1991.
 109. Bednar, J. and T. L. Watt, "Alpha-trimmed Means and Their Relationship to Median Filters", *IEEE Transactions Acoustics, Speech, and Signal Processing*, Vol. 32, No. 1, pp. 145–153, 1984.
 110. Ferman, A. M., A. M. Tekalp and R. Mehrotra, "Robust Histogram Descriptors for Video Segment Retrieval and Identification", *Submitted to IEEE Trans. on Image Processing*, 2000.
 111. Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992.
 112. Swain, M. J. and D. H. Ballard, "Color Indexing", *International Journal of Computer Vision*, Vol. 7, No. 11, pp. 11–32, 1991.
 113. Johnson, R. A. and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall, 1998.
 114. Erdem, C. E., A. M. Tekalp and B. Sankur, "Non-rigid Object Tracking with Feedback of Performance Evaluation Measures", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 9-14 Dec., Hawaii, USA, 2001.

115. Canny, J., “A Computational Approach to Edge Detection”, *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 8, No. 6, pp. 679–698, 1986.
116. Mortensen, E. N. and W. A. Barrett, “Interactive Segmentation with Intelligent Scissors”, *Graphical Models and Image Processing*, Vol. 60, pp. 349–384, 1998.
117. Duda, R. O. and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, 1973.
118. Kass, M., A. Witkin and D. Terzopoulos, “Snakes: Active Contour Models”, *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321–331, 1988.
119. Amini, A. A., T. E. Weymouth and R. C. Jain, “Using Dynamic Programming For Solving Variational Problems in Vision”, *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 12, No. 9, pp. 885–867, 1990.
120. Zhang, J. and J. W. Modestino, “Model-fitting Approach to Cluster Validation with Application to Stochastic Model-based Image Segmentation”, *IEEE Transactions Pattern Analysis, and Machine Intelligence*, Vol. 12, 1990.
121. Borgefors, G., “Distance Transformations in Digital Images”, *Computer Vision, Graphics, and Image Processing*, Vol. 34, pp. 344–371, 1986.
122. Lucas, B. D. and T. Kanade, “An iterative Image Registration Technique with an Application to Stereo Vision”, *Proc. DARPA Image Understanding Workshop*, pp. 121–130, 1981.