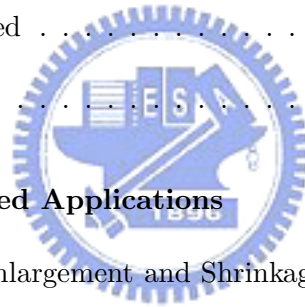


# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Existing Methods of Video Segmentation . . . . .	2
1.2	Main Contributions of This Work . . . . .	5
1.3	Organization of the Thesis . . . . .	6
<b>2</b>	<b>A Region-Based Video Segmentation Algorithm</b>	<b>8</b>
2.1	The Segmentation Method . . . . .	9
2.2	Seed-Area Identification . . . . .	10
2.3	Initial Segmentation . . . . .	14
2.3.1	Strategies in Region Growing . . . . .	15
2.3.2	Region Merging . . . . .	19
2.4	Motion-Based Region Integration . . . . .	23
2.5	Region Tracking and Updating . . . . .	25
2.6	Experimental Results . . . . .	27
<b>3</b>	<b>An Enhanced Region-Based Video Segmentation Algorithm with Extrac- tion of Overlaid Objects</b>	<b>35</b>
3.1	Brief Review and Method Introduction . . . . .	36

3.2	Motion Analysis in Pixel Tier: Motion Estimation and Change Detection . . .	40
3.2.1	Motion Estimation . . . . .	40
3.2.2	Detection of Changed Pixels . . . . .	41
3.3	Edge Analysis in Pixel Tier: Edge Detection and Patch Delineation . . . . .	44
3.4	Foreground Tier: Motion-Oriented Foreground Separation . . . . .	47
3.4.1	Foreground Sketch . . . . .	48
3.4.2	Foreground Extraction . . . . .	49
3.4.3	Foreground Validation . . . . .	49
3.5	Overlays Tier: Segmentation of Overlaid Objects . . . . .	50
3.6	Experimental Results . . . . .	53
<b>4</b>	<b>Edge-Based Morphological Processing for Efficient and Accurate Video</b>	
	<b>Object Extraction</b>	<b>63</b>
4.1	Background . . . . .	64
4.2	Existing Methods . . . . .	65
4.3	The Proposed Method . . . . .	68
4.3.1	Mask Sketch . . . . .	69
4.3.2	Mask Refinement . . . . .	71
4.3.3	Remarks on Complexity . . . . .	72
4.4	Experimental Results . . . . .	73
<b>5</b>	<b>Reduced-Complexity Motion Analysis and Edge Processing for Accurate</b>	
	<b>Identification of Object Boundaries</b>	<b>84</b>
5.1	Motion-Related Analysis . . . . .	86
5.1.1	Forward Tracking . . . . .	86

5.1.2	Backward Validation . . . . .	89
5.2	Mask Refinement . . . . .	89
5.2.1	Mask Rounding . . . . .	91
5.2.2	Footprint Tightening . . . . .	93
5.2.3	Boundary Sharpening . . . . .	94
5.2.4	Mask Tuning . . . . .	97
5.2.5	Summary and Comparison with Other Edge-Based Methods . . . . .	98
5.3	Experimental Results . . . . .	99
5.3.1	Illustration of Algorithm Steps . . . . .	100
5.3.2	Algorithm Performance . . . . .	104
5.3.3	Algorithm Speed . . . . .	109
5.4	Discussion . . . . .	110
<b>6</b>	<b>Support of Content-Based Applications</b>	<b>112</b>
6.1	Methods for Object Enlargement and Shrinkage . . . . .	112
6.2	Experimental Results and Discussions . . . . .	116
<b>7</b>	<b>Conclusions and Future Work</b>	<b>120</b>



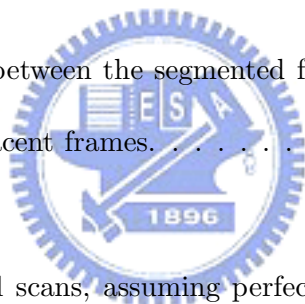
# List of Figures

1.1	A typical structure of the video segmentation system. . . . .	3
2.1	System structure of the video segmentation. . . . .	10
2.2	Illustration that a single global threshold may not capture local intensity structures property. Above: a possible image intensity profile; below: corresponding gradient amplitude. . . . .	12
2.3	Illustration that a more local gradient intensity analysis may capture the more detailed structures in an image. Above: looking at two image sections separately; below: separate intensity analysis of the two sections reveals detailed image structure in each section. . . . .	12
2.4	Arbitrary example for illustration of seed-area identification. (a) Bright and dark region sets, $R_{hj}$ indicates separated regions. (b) Separately treated each $R_{hj}$ to obtain seed areas (dark regions): seed set $S = \{s_k\}$ . . . . .	13
2.5	Illustration of an example of region growing. . . . .	16
2.6	Illustration of region growing. . . . .	17
2.7	Illustration of the final result of Fig. 2.5. . . . .	18

2.8	Arbitrary example for illustration of the region-map and its corresponding RAG. (a) An arbitrary region-map example. (b) Corresponding weighted RAG. (a) Assume regions in (a), in ideal, become two regions after refining out small regions. (b) Weighted RAG. . . . .	21
2.9	Illustration of region merging and RAG changing: Merging Step-by-step. . .	22
2.10	Illustrative temporal progression of different regions. . . . .	26
2.11	Segmentation performance of the Table Tennis Sequence. (a) Original frame 1. (b) Regions by bright and dark segmentation. (c) Seed (bright areas) and uncertain regions (dark areas). (d) Initial segmentation result of the (a). (e) Background portion found in the second frame. (f) Uncovered area in frame 2 (dark points). . . . .	28
2.12	Tracking of ping-pong ball. . . . .	29
2.13	Ball-playing-arm tracking result. . . . .	30
2.14	Recreated frame 1 after removing ping-pong ball and player's left hand. . . .	30
2.15	VO tracking results of Claire. . . . .	31
2.16	VO tracking results of Akiyo. . . . .	32
2.17	Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask. . . . .	33
2.18	Fractional agreement between the segmented foreground object masks of the Akiyo sequence in adjacent frames. . . . .	33
3.1	Illustration of video scene with overlaid objects or region showing different motion. . . . .	38
3.2	Real-life examples where video scene contains multiple overlaid objects or regions showing different motion. . . . .	39

3.3	Overall structure of the proposed algorithm. . . . .	40
3.4	Filtering masks for detecting moving pixels. . . . .	43
3.5	Reducing covered and uncovered areas in change detection. . . . .	45
3.6	Patch delineation algorithm. . . . .	46
3.7	Process of initial patch delineation. . . . .	46
3.8	Structure of the foreground separation algorithm. . . . .	47
3.9	Structure of object segmentation algorithm. . . . .	50
3.10	Some patch delineation results for Akiyo, Salesman, and Flower Garden. (a),(c), and (e) are the originals, and (b), (d), and (f) are the results, where the dif- ferent patches are distinguished by different gray levels. . . . .	54
3.11	Some change detection results for Akiyo, Salesman, and Flower Garden. Num- ber below each picture gives the frame number. In the case of Akiyo and Salesman, the changed pixels are marked in white, but for clarity, in the case of Flower Garden the unchanged pixels are instead marked in white. . . . .	55
3.12	Foreground region of Akiyo. Number below each picture gives the frame number.	56
3.13	Foreground region of Salesman. Number below each picture gives the frame number. . . . .	56
3.14	Foreground region of Flower Garden. Number below each picture gives the frame number. . . . .	57
3.15	Object “right arm” obtained in the overlays tier from the foreground region of Salesman. Number below each picture gives the frame number. . . . .	57
3.16	Object “head” obtained in the overlays tier from the foreground region of Salesman. Number below each picture gives the frame number. . . . .	58

3.17	Object “Tree trunk” obtained in the overlays tier from the foreground region of Flower Graden. Number below each picture gives the frame number. . . .	58
3.18	Object “house right” obtained in the overlays tier from the foreground region of Flower Graden. Number below each picture gives the frame number. . . .	59
3.19	Synthesized Flower Garden scene from segmentation result, with tree removed. Upper row: original frames 10, 20, 30; middle row: synthesized frames. . . .	59
3.20	Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask. . . . .	60
3.21	Fractional agreement between the segmented foreground object masks of the Akiyo sequence in adjacent frames. . . . .	61
4.1	Example of orthogonal scans, assuming perfect edge detection that finds all the boundary pixels of the object. (a) Object shape. (b) Resulting mask of maximal scan. . . . .	66
4.2	Another example of orthogonal scans, assuming perfect edge detection that finds all the boundary pixels of the object. (a) Object shape. (b) Resulting mask of minimal scan. (c) Resulting mask of maximal scan. . . . .	67
4.3	Results of minimal scan when the lower-right object edge is missing. (a) Object of Fig. 4.1(a). (b) Object of Fig. 4.2(a). . . . .	68
4.4	The proposed method of edge linking. . . . .	69



4.5	Arbitrary example for illustration of the proposed algorithm. (a) CDM. (b) Result of CDM round-out. (c) Result of CDM round-out, with edge pixels therein marked in black. (d) Result of segmental row scans. (e) Result of segmental column scans, i.e., result of boundary tightening. (f) Result of boundary tightening with edge pixels in the mask marked in black while other pixels marked in gray. . . . .	77
4.6	Illustration of the mask refinement method. (a) Zoomed-in section of the mask sketch result for illustration use. (b) After stopping of one-pixel gaps. (c) Respective search areas of shortest-path algorithm for edge discontinuities ( $a, b$ ) and ( $c, B$ ). (d) Final result of edge linking between $A$ and $B$ . . . . .	78
4.7	Illustration of algorithm behavior. (a) An original frame of Mother-and-Daughter sequence. (b) CDM. (c) Rounded CDM. (d) Result of mask sketch. . . . .	79
4.8	Result of mask refinement at different search bandwidths. (a) $D_w = 2$ . (b) $D_w = 5$ . . . . .	79
4.9	Segmentation results for Mother and Daughter based on maximal scan. (a) Result of maximal scan on CDM. (b),(c),(d) Results of mask refinement with $D_w = 20, 60,$ and $100,$ respectively. . . . .	80
4.10	Result of mask refinement based on minimal scan at different search bandwidths. (a) $D_w = 2$ . (b) $D_w = 5$ . . . . .	80
4.11	Comparison of algorithm efficiency and accuracy in segmentation of Mother-and-Daughter sequence. (a) Maximal-scan based processing. (b) Minimal-scan based processing. (c) Proposed method. . . . .	81
4.12	Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask. . . . .	82



4.13	Fractional agreement between the segmented foreground object masks of the Akiyo sequence in adjacent frames. . . . .	83
5.1	Structure of the proposed algorithm. . . . .	85
5.2	Object-based motion estimation. (a) An object under consideration (shaded region) in the previous frame has moved to a different position in the current frame (dashed contour). Each small square is $BW \times BW$ in size. (b) Illustrating the idea that motion estimation is carried out on the $3 \times 3$ macroblock centered at the treated block. . . . .	87
5.3	Mask refinement procedure. . . . .	91
5.4	An arbitrary example for illustration of the mask refinement method. (a) The coarse mask (gray pixels). (b) The coarse mask with the edge pixels therein marked in black. . . . .	92
5.5	Illustration of the mask rounding procedure. (a) The mask after orthogonal scans and taking the union. (b) The eroded mask ( $FG_r$ ). . . . .	92
5.6	Illustration of the footprint procedure. (a) $FG_r$ with the edge pixels therein marked in black. (b) Row map. (c) Column map. (d) The footprint-tighten mask ( $FG_t$ ), with edge pixels therein marked in black. . . . .	93
5.7	Illustration of the boundary sharpening procedure. (a) Mask after overgrowing pruning in the horizontal direction. (b) Mask after overgrowing pruning in the vertical direction.. (c) A section of the horizontally pruned mask. (d) The same section after edge filling in the horizontal direction. (e) Converged result of overgrowth pruning and edge filling in the horizontal direction. (f) Converged result of overgrowth pruning and edge in the vertical direction. . . . .	95

5.8	Illustration of the mask tuning procedure. (a) Intersection of the two directional masks in $FG_s$ . (b) $O_{i,n}$ . . . . .	98
5.9	Illustration that more than basic morphological operations are needed on intersection map of edge-based orthogonal scan to obtain accurate object mask. (a),(b) Results of horizontal and vertical scan, respectively. (c) The intersection map. . . . .	100
5.10	Illustration of the mother-related analysis in the proposed algorithm using frame 74 of the Mother and Daughter sequence as example. (a),(b) Original frame 73 and 74, respectively. (c) Foreground object $O_{1,73}$ segmented from frame 73. (d) Change detection result $CD_{74}$ . (e) Union of $P_{1,74}$ and $CD_{74}$ . (f) Coarse mask $PCD_{1,74}^B$ . . . . .	101
5.11	Illustration of the mask refinement procedure in the proposed algorithm using frame 74 of the Mother and Daughter sequence as example. (a) Union mask of two orthogonal scan results. (b) Result of mask rounding, $FG_r$ . (c),(d) Row and column maps obtained in footprint tightening, respectively. (e) Result of footprint tightening, $FG_t$ . (f),(g) After one iteration of boundary sharpening in horizontal and vertical directions, respectively. (h) Intersection map of $FG_s$ .	103
5.12	Example segmentation results of the Akiyo sequence. Left, top to bottom: Original frame, 30, 70, and 117. Right: segmentation results. . . . .	105
5.13	Example segmentation results of the Mother and Daughter sequence. Left, top to bottom: Original frame, 50, 95, and 120. Right: segmentation results. . .	106
5.14	Example segmentation results of the Salesman sequence. Left, top to bottom: Original frame, 10, 20, and 30. Right: segmentation results. . . . .	107

5.15	Example segmentation results of the Foreman sequence. Left, top to bottom: Original frame, 4, 24, and 34. Right: segmentation results. . . . .	108
5.16	Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask. . . . .	109
5.17	Fractional agreement between the segmented foreground object of the Akiyo sequence in adjacent frame. . . . .	110
6.1	Amplitude response of the MPEG-2 enlargement and shrinkage filters. . . . .	114
6.2	Proposed enlargement algorithm, illustrated by example. (a) Enlargement by two or three times, (b) Enlargement by two, four, or five times (Gray squares denote pixels in the original object mask or as obtained by interpolation us- ing the MPEG-2 filter; white squares denote linear linear interpolated pixels. Dashed lines indicate linear interpolation between nearest pixels.) . . . . .	115
6.3	Proposed Shrinkage algorithm, illustrated by example. (a) Shrinkage by two or three times, (b) Shrinkage by two, four, or five times (Gray squares denote pixels in the original object mask or as obtained by decimation using the MPEG-2 filter; white squares denote linear linear interpolation pixels. Dashed lines indicate linear interpolation between nearest pixels.) . . . . .	116
6.4	Similarity, in PSNR, between the original segmented video object in CIF Mother- and-Daughter sequence and that enlarged X times and shrunk to original size, where X is between 2 and 5. (a) For interior of object (inside a three-pixel- wide band at object boundary). (b) For the three-pixel-wide band at object boundary. . . . .	118
6.5	Synthesized scenes using segmented (and shrunk by 2 times) video objects and other contents. . . . .	119

6.6 Applications from superposed VOs. (a) Motion history of the ball in the Table-  
tennis. (b) VO in the Mother-and-Daughter fuse a statue for exhibition. . . . 119



# List of Tables

2.1	Sorted List of Adjacent Pixels for the Two Seed Regions . . . . .	18
2.2	Sorted List of Adjacent Pixels for the Two Seed Regions After Merging (3,2)	18
2.3	Sorting the RAG's Weights . . . . .	23
2.4	Average Computing Time in ms per Frame of Major Algorithm Functions . .	34
3.1	Average Computing Time in ms per Frame of Major Algorithm Functions . .	61
4.1	Average Computing Time in ms per Frame of Major Algorithm Functions . .	76
5.1	Number of Pixels Deleted in Each Iteration of Boundary Sharpening for the First Frame . . . . .	102
5.2	Average Computing Time in ms per Frame of Major Algorithm Functions . .	111

# Chapter 1

## Introduction

As digital data gradually enter people's lives, consumers wonder when digital images and video will become accessible everywhere. Consumers desire access digital contents because an image or video affects human perceptions more directly than any other medium. "An image or video can speak more than a thousand words."

Video information is quite large in size, and needs to be compressed for efficient transmission and storage. Some coding technologies, such as MPEG-1, MPEG-2, H.261 and H.263, have led to the quick development of multimedia applications. The coding techniques can be used for multimedia storage, videophones, videoconferencing, high-definition television (HDTV), video-on-demand (VOD), and video over wireless networks. These coding methods treat each frame in video as fixed array of pixels and partition each input frame into square or rectangular blocks as the elementary coding unit, and they have been called frame-based coding methods.

In contrast to the frame-based methods, MPEG-4 and MPEG-7 allow content based processing. the MPEG-4 video standard was designed for high efficiency, very low bit-rates manipulation, and universal access. MPEG-7 is for multimedia content description interface,

which describes contents of video scenes for effective retrieval and searching [14, 39, 43]. The content based methods yield more flexibility in video applications [7, 12, 15, 28, 29, 30, 32, 44, 59]. Subjectively meaningful video contents, such as people, houses, hands, and cars, have been called “video objects” or “semantic objects.” Segmented objects can be separately processed for various uses [46, 47]. Thus video segmentation is a key technique to fully exploit the potential of MPEG-4 and MPEG-7. Although MPEG-4 and MPEG-7 have been developed, neither standard regularizes video segmentation methods. In addition, the video segmentation schemes available in the literature still leave room for improvement.

## 1.1 Existing Methods of Video Segmentation

Automatic video segmentation refers to segmenting images into semantic regions. Automatic segmentation of moving objects (foreground) from the background aims to generate a VOP (Video Object Plane, in MPEG-4 technology). Many video segmentation approaches have been proposed [9, 10, 16, 17, 26, 27, 34, 35, 36, 37, 38, 54, 56, 57, 58]. The approaches can be classified into motion-based segmentation (using only motion information) [9, 38, 56, 57] and spatio-temporal segmentation (combining of temporal and spatial segmentation) [10, 16, 17, 26, 27, 34, 35, 36, 37, 54, 58]. Pure motion-based approaches suffer from inaccurate object boundaries due to that motion estimation is an ill-posed problem. Spatio-temporal segmentation approaches can be further classified into region-based and edge-linking-based methods. Spatio-temporal methods appear to attain the best boundary accuracy. Fig. 1.1 summarizes the structure of typical video segmentation.

Wang and Adelson [56, 57] published the first study on motion segmentation. They discussed motion models and orientation consistencies rather than spatial information, which results in boundary inaccuracy. Borshukov *et al.* [4] enhanced Wang’s motion selection method

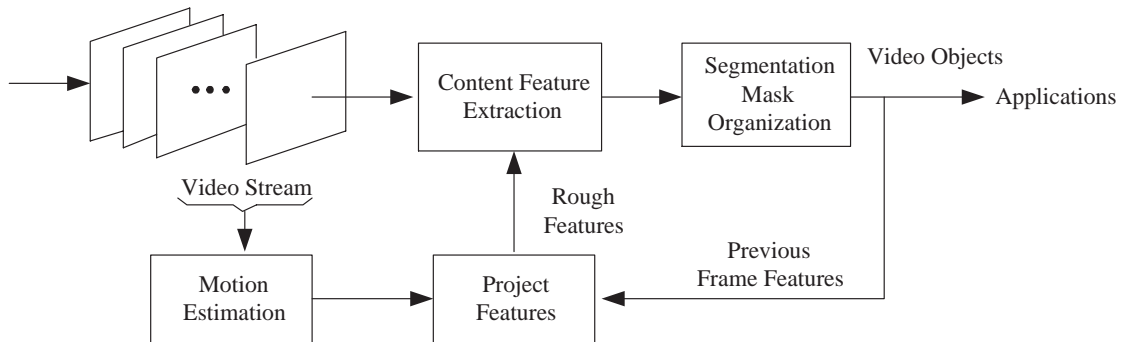


Fig. 1.1: A typical structure of the video segmentation system.

to consider homogeneity of motion orientation. Their study aimed only to improve Wang's scheme, and did not consider spatial information. Neri *et al.* [38] concentrated on statistical models to extract moving objects, creating mathematical models under the assumption of inter frame motion. The extraction process lacked the spatial-border information, and therefore object boundaries were inaccurate. However, Neri *et al.* applied a morphological operation to fill in the inside-holes of the objects. Some recent studies only discuss detecting the moving areas for particular purposes (e.g., medical sonograms, infrared imagery detection, security and traffic surveillance). Detecting a video's moving areas is important in practical use, and its post processing significantly impacts segmentation performance [9, 27, 35, 37]. Chien *et al.* [9] eliminated shadow noise and foreground noise by a highly complex motion estimation method for tracking.

Spatial-temporal approaches are very attractive for automatic image segmentation since they efficiently handle spatial descriptions such as area, size and connectivity, which can be viewed as region-based features. Temporal information is then applied to carry these features into the next frames. Wang's spatial-temporal scheme [58] centered on morphological operations to unify spatial data for straightforward segmentation, using a watershed algorithm.



The algorithm was also modified for region updates performed on zones that showed changes between frames. Gao *et al.* [16] proposed a method considering color information similar to Wang's scheme.

Wang's method, like many others, is slow. Chien modified the method to improve its speed, but this improvement did not apply to the first frame. Kim *et al.* published a paper [26] that further concentrated on change detection and developed foreground extraction over segmented zones, which yielded objects with accurate boundaries. However, the temporal detection sometimes fails on the inside of a zone, causing objects to be drawn incompletely. Tsaig and Averbuch [54] modeled each zone's object-based motion information using a probability model (i.e., the Markov random field, MRF).

This study applies a spatio-temporal method to consider the automatic segment a natural video scene automatically into a composition of objects. Automatic video segmentation methods can be classified into three categories: heuristic, probabilistic, and mixed. A probabilistic method models the video frames as two-dimensional random processes and attempts to maximize some probabilistic measure of goodness of segmentation [61], while the heuristic scheme analyzes the motion and texture to split the video frames into regions with disparate heuristic motion and texture features [34]. Probabilistic methods frequently involve an iterative procedure initialized with the result of a heuristic segmentation method. The mixed algorithms employ heuristic analysis in one part of the algorithm and probabilistic optimization in another part. An example of the probabilistic approach can be found in [51], and some examples of the mixed approach are can be found in [42] and [54]. This study considers the heuristic approach.

One common video segmentation approach is described briefly as follows. First, a frame (or some key frames) is selected as the starting frame to execute segmentation. Then each

object's features, such as color, shape, motion and texture are analyzed. Then, each object is mapped to next frame. Then, inter-frame segmentation is applied to each subsequent frame following the previous frame. Inter-frame segmentation automatically tracks each object, and the resulting partition can achieve the temporal coherence of object labeling, i.e., object correspondence, and maintain segmentation similarity between successive frames. Therefore, more reliable segmentation results can be attained when the feature alignment and repeated feature aggregation are robust.

Some video segmentation methods employ change detection to separate moving objects from stationary background. However, objects obtained from change detection may contain inside holes and boundary inaccuracy. Therefore, inside patching and boundary accuracy are two important issues.

## 1.2 Main Contributions of This Work



The main contribution of this study in video segmentation is summarized as follows.

1. The method linking region-based spatial segmentation and temporal analysis is developed. This method combines intensity region growing, motion analysis and tracking techniques, and it is shown that the method is sufficient to determine a complete segmentation. The complexity of this algorithm is similar to [10] and [58]. Subjectively, this method can obtain better results.
2. A multi-tiered segmentation algorithm was developed to tracks foreground objects and slice up into different objects by motion layers. This method extends region-based segmentation [58]. The integration with a change detection scheme is novel, and an overlays segmentation can be made over the changed areas.

3. A novel morphological filtering is proposed for efficient and accurate video object extraction. This method integrates morphological filtering and a shortest-path algorithm. This method can significantly improve the complexity of [37], and the object boundary accuracy is better than [27].
4. An Algorithm for fast object boundary correction were created. This algorithm is based on two novel designs: an edge-linking method for object extraction and a complexity reduction for motion estimation. The approach can accurately obtain segmentation and robustly propagate the objects between frames. The complexity of this method can be significantly reduced.

### 1.3 Organization of the Thesis

This dissertation is organized as follows.

Chapter 2 develops a method for automatic segmentation which is based on low-level and spatial segmentation and temporal analyses. Region based spatial segmentation is introduced, including region growing and region merging techniques, and then a complexity reduction of motion analysis and object tracking are presented. This chapter shows that the method is sufficient to determine a complete segmentation. The approach using region features and temporal analysis is robust for accurate segmentation.

Chapter 3 presents a spatio-temporal algorithm for multi-tier structure segmentation and tracking. It is based on analysis of apparent motion, inter-frame pixel value changes, edges, and textural homogeneity of image regions. It is designed to be able to separate multiple overlaid objects that do complicated or relatively fast motion, to handle object deformation, and to address appearance and disappearance of objects. The above abilities distinguish the

algorithm from some recently published ones. Experiments on several different kinds of video show that the algorithm can yield reasonably good identification of object boundaries.

Chapter 4 considers the edge-linking approach for accurate locating of moving object boundaries in video segmentation. We review the existing methods and propose a “segment-based” scheme designed for efficiency and better accuracy. The algorithm is based on morphological filtering and designed with computational complexity also in mind. Experiments show that the scheme yields good performance.

Chapter 5 discusses a design of the edge-linking approach for fast processing. The way is a “pixel-based” method designed for efficiency and better accuracy. Same mind of the chapter 4, the algorithm is designed with computational complexity as well. In addition, a reduced-complexity motion analysis is presented. A comparison with some other motion estimation is also addressed. Experiments show that the scheme yields good performance and fast processing.

Chapter 6 investigates some tools for applications by the integration of VOs and nature images. Some simply new methods for shape are proposed.

Finally Chapter 7 gives the conclusion and points out some topics for potential future work.

## Chapter 2

# A Region-Based Video

# Segmentation Algorithm



This chapter describes a technique for region based video segmentation.\* Methods are presented for generating regions, competitive expansion of regions, integrating regions and tracking regions. As we can see, all these are region-based processing. In generating of regions, we select the region-growing method for easy implementation. The system studied in this chapter introduces the concept of competitive seed areas. We first describe a modified method suitable for nature scene uses. Region growing is also used for subsequent frames. We then describe region growing that is based on competitive characters. Then regions are determined different in the first and the subsequent frames. Then a simplified motion estimation is introduced to track the region set. Additionally, this study considers region integration

---

\*Part of this chapter has appeared in Yih-Haw Jan and D. W. Lin, "A method for video segmentation based on object tracking," in *Proc. Int. Symp. Commun.*, Nov. 2001, paper 10.4. and Y.-H. Jan and D. W. Lin, "Image sequence segmentation via heuristic texture analysis and region tracking," in *SPIE* vol. 4671, in *Visual Commun. Image Processing*, pt. 2, Jan. 2002, pp. 543–551.

employing motion information. The final section exhibits the segmentation results.

## 2.1 The Segmentation Method

This section outlines an automatic region-based segmentation method. Fig. 2.1 shows the proposed video segmentation method. The proposed method comprises four tasks, namely, seed-area identification, initial segmentation, motion-based segmentation, and region tracking and updating.

Seed-area identification is performed on every input frame. Based on simple intensity analysis, a number of relatively homogeneous “seed areas” in the frame are identified for use in subsequent image segmentation and region tracking. The initial segmentation is performed on the first frame of the sequence only. Beginning with the seed areas, a segmentation of the first frame is achieved via regional growth and merging, where the region growth procedure attempts to locate the region boundaries where the local image texture changes significantly. The other two tasks are applied to all subsequent frames. The task of motion-based segmentation extracts the moving regions by estimating the motion of each segmented region and integrates the regions exhibiting similar motion. The task of region tracking and updating projects each moving region onto the next frame based on the dense motion vectors obtained in the last task, validates the mapping by examining the seed areas involved, and re-segments the uncovered areas, the overlapped areas, and other areas in the next frame which exhibit undesirable features. Clearly, the proposed method contains only one pass which operates in the forward direction. Without higher-level intelligence, a one-pass, forward-only method generally has difficulty achieving correct object segmentation during the first few frames of the appearance of a new object. The mechanism incorporates tasks of motion-based segmentation and region tracking and updating that can modify the segmentation by

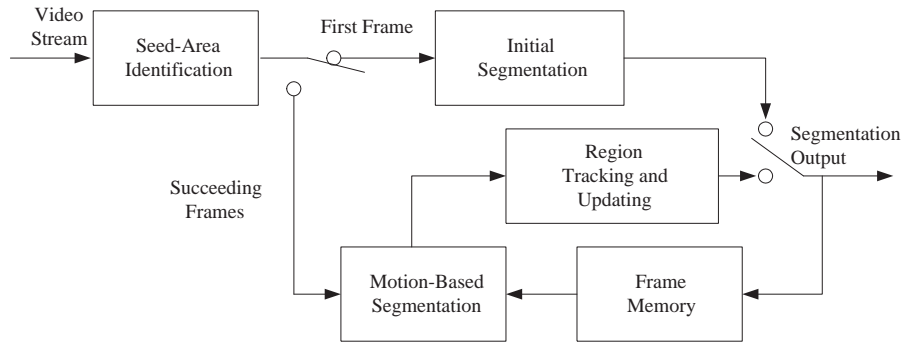


Fig. 2.1: System structure of the video segmentation.

adjusting the object boundaries in later frames. Each task is detailed below.

## 2.2 Seed-Area Identification

In this section, we discuss initial classification that capitalizes on the homogeneity properties and analysis of gradients to identify object boundaries. Watershed approaches [17, 34, 58] and histogram thresholding [41] are among the most commonplace methods. These methods generally make use of homogeneity valleys over an image and cluster them in some way. Regional growth can be considered belonging to the broad watershed methods and enjoys the advantage of easy implementation. This approach considers intensity similarity relative to spatial locations. Therefore each segment is a closed region and has homogeneous features, because regional growth begins with valleys, then adds adjacent pixels to regions where pixels show similar features.

Since region growth is used for initial classification, in practice there are two main issues related to region growth. These problems are as follows: first, how are the geometric valleys selected? and second, how should the similarity criteria be?

The purpose of this task is to divide a frame into a number of seed areas for use in

subsequent image segmentation and region tracking. The rationale behind the particular approach is illustrated in Figs. 2.2 and 2.3.

Often, an image contains patches wherein the intensity (and color) values are relatively homogeneous. Still-image segmentation often capitalizes on such property and analyzes intensity gradients to determine boundaries. Incidentally, the watershed approach represents such thinking. Since our segmentation method assumes no high-level knowledge concerning the image contents but employs only low-level signal processing, it is natural to base the initial segmentation on intensity analysis. And it is natural to consider thresholding the intensity gradients so that gradients above the threshold are considered to mark region boundaries. However, it may not be appropriate to use a single global threshold for the entire image, because the intensity structures of different areas of the image may be different. Using a single global threshold may miss some finer local intensity structures. Figs. 2.2 illustrates the situation where areas A and B are both composed of two patches having relatively homogeneous intensities within each patch but significantly different intensities between patches. But a single-threshold gradient-based segmentation does not find this out. Therefore, the intensity analysis should be conducted more locally. On the other hand, since overly small objects are usually not considered appropriate, the intensity analysis should avoid producing overly fine segmentations. Fig. 2.3 illustrates the idea that a more local gradient analysis may capture the more detailed structures in an image.

In ways, first calculate the mean image intensity  $\mu_I$  of the whole frame. The frame is termed a bright image if more than 50% of the pixels have intensity values exceeding  $\mu_I$ ; otherwise it is considered a dark image. The pixels with intensity values exceeding  $\mu_I$  are termed bright pixels, and those with intensity values below  $\mu_I$  are termed dark pixels. The following describes the operations conducted on a bright image. The operations performed



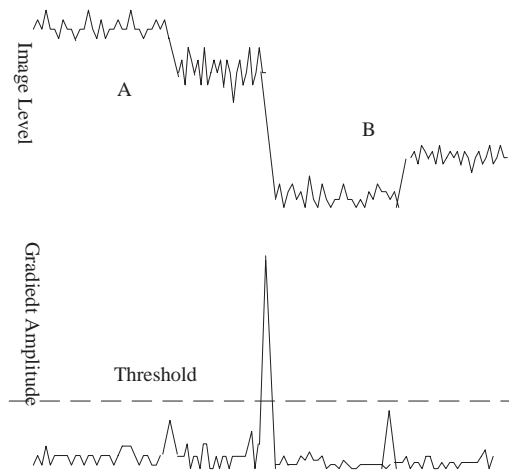


Fig. 2.2: Illustration that a single global threshold may not capture local intensity structures property. Above: a possible image intensity profile; below: corresponding gradient amplitude.

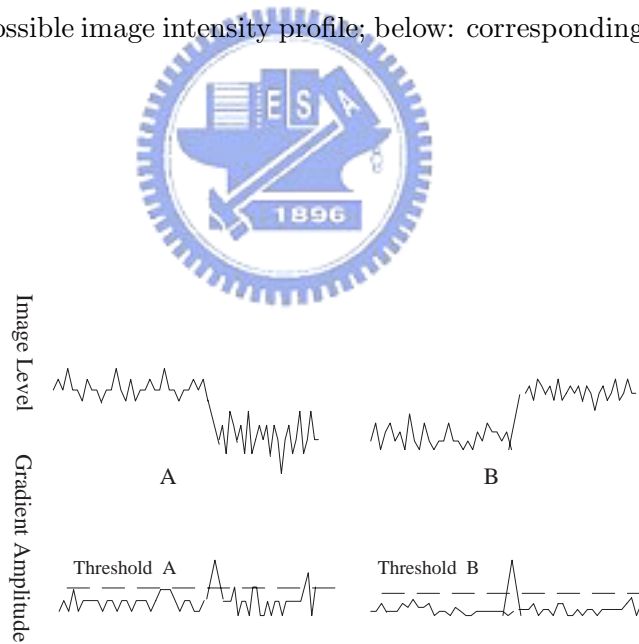


Fig. 2.3: Illustration that a more local gradient intensity analysis may capture the more detailed structures in an image. Above: looking at two image sections separately; below: separate intensity analysis of the two sections reveals detailed image structure in each section.

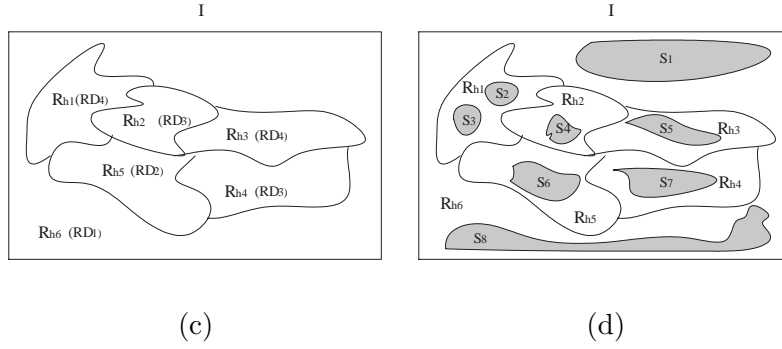


Fig. 2.4: Arbitrary example for illustration of seed-area identification. (a) Bright and dark region sets,  $R_{h_j}$  indicates separated regions. (b) Separately treated each  $R_{h_j}$  to obtain seed areas (dark regions): seed set  $S = \{s_k\}$ .

on a dark image are complementary.

For a bright image, each connected set of bright pixels is identified and rank-ordered based on descending size. Let  $R_{hi}$  denote the  $hi$ -th the bright region. The intensity mean  $\mu_{R_{hi}}$  and intensity variance  $\sigma_{R_{hi}}^2$  of this region are calculated. A region growing process is performed starting with the largest bright region. The neighboring points with intensity values exceeding  $\mu_{R_{hi}} - \sigma_{R_{hi}}$  are merged into the region. Some small bright regions can be relegated to the dark background. The remaining, unmerged dark pixels then may form multiple connected regions. Consequently, all regions are identified.

For each region in the final set, either bright or dark, the gradient structure is analyzed and one or more areas of low gradient value are identified to form the desired seed areas. This approach achieves the localized gradient analysis illustrated in Fig. 2.3. Each set of such pixels then constitutes a seed area, termed  $S$ , which is used in subsequent image segmentation and region tracking.

Fig. 2.4 illustrates an implement related to the initial classification and the local gra-

dient analysis. Fig. 2.4(b) shows the initial classification region set, and the regions,  $R_{hj}$ , are labeled by scanning  $RD_x$ . Fig. 2.4(b) shows the formation of seed areas  $S = \{s_k\}$  by considering each  $R_{hj}$ . Moreover,  $S$  awaits further evolution from the region growth process.

## 2.3 Initial Segmentation

The task is achieved in two steps: region growing and region merging. Region growing involves growing the seed areas by examining both the local variation and area-wide variation in pixel intensity values. Meanwhile, region merging merges the small regions until they all reach a minimum specified size.

To improve the regional growth segmentation process, various methods were devised to select the homogeneity criterion automatically from the image. This study employs a method based on the previously developed method of seed-area identification for locating seed regions that focus especially on local seeds. As seed regions are chosen, each routine grows the highest similar intensity parameter pair of the region and its unseeded neighbor pixel. The following steps comprise the basic process of region growth:

The following steps are the universal processes in region growing:

- (1) Select the seed pixel  $S_r$ .
- (2) Check the adjacent pixels,  $p_r$ , and then add  $x$  to  $S_r$  if  $x$  is the most likely pixel to  $S_r$  with cost  $C_{s_r}$ , i.e.,  $\min_{x \in p_r} C_{s_r}(x)$ .
- (3) Repeat (2), until no more pixels can be added.

In the second step, namely, region merging, the size of each region is made to at least equal a predefined value  $N_r$ .

### 2.3.1 Strategies in Region Growing

The goal of region growing is to partition the image into a set of disjoint regions. Region-growing considers the similarity of intensity relative to spatial locations. Region-growing is based on the step of taking some pixels, called seeds, and growing the regions around them based upon certain homogeneity criteria. If the adjoining pixels resemble the seed, they are merged with the seed within a single region. The process continues until all the pixels in the image are assigned to the most similar region of each.

It is one of the conceptually simplest methods, however, for the use in practice, we must consider its complexity, which is related to the definition of seed regions and the similarity rules. Researches in region-growing mostly concerns the measure of distance to improve the segmentation effect.

The measure of similarity here uses features such as region mean and variance of intensity. More specifically, in region growth, each seed area is assigned a distinct label if its area exceeds a certain threshold, say  $N_r$ . Meanwhile, other pixels are merged into the seed areas individually via the following process. Let  $s_k$  denote the  $k$ -th seed area. A hypothetical example involves  $s_k$  developing to  $R_k$ . Its mean  $\mu_{R_k}$  and variance  $\sigma_{R_k}$  in intensity are calculated. Furthermore, the local texture information at each pixel represented by the local mean and variance in intensity in a  $3 \times 3$  window are calculated. For a pixel  $p_{i,j}$  at location  $(i, j)$ , the local mean and variance are denoted  $\mu_{i,j}$  and  $\sigma^2(i, j)$ , respectively. This is done for all the pixels located immediately outside the border of each seed area are calculated. the “distance” between one such pixel  $p_{i,j}$ , and the seed area  $R_k$  is calculated as

$$d(R_k, p_{i,j}) = \alpha|I(i, j) - \mu_{R_k}| + \beta|\bar{\sigma}_{R_k} - \sigma(i, j)| + \gamma|I(i, j) - \mu(i, j)| \quad (2.1)$$

where  $I(i, j)$  is the intensity of  $p_{i,j}$ ,  $\bar{\sigma}_{R_k}$  is the average of the local standard deviations of all

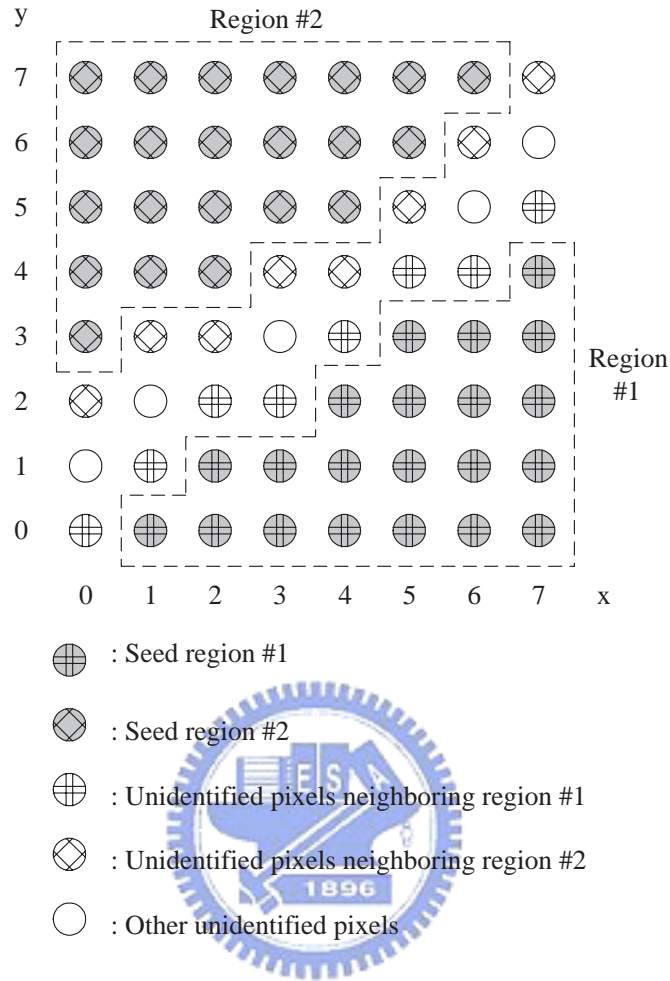
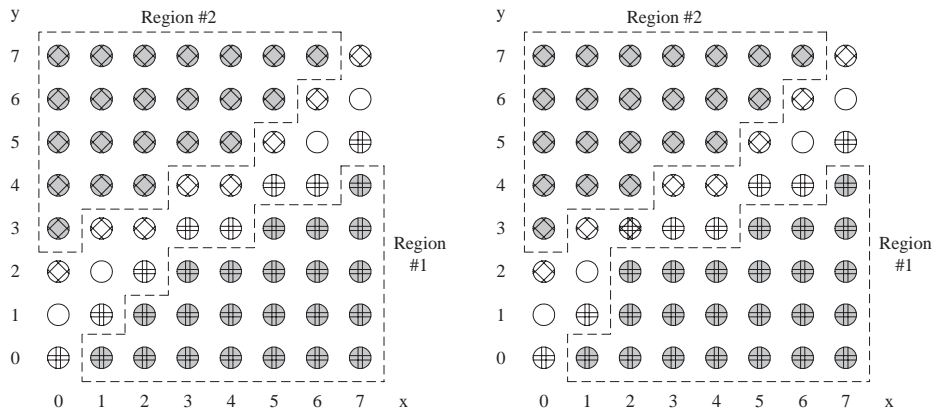


Fig. 2.5: Illustration of an example of region growing.

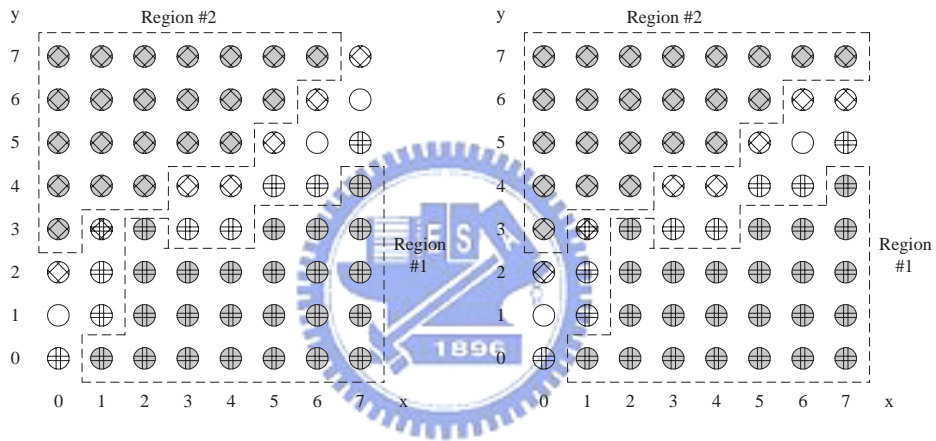
the pixels in  $R_k$ , and  $\alpha$ ,  $\beta$ , and  $\gamma$  are some weighting factors. From all the pixel-area pairs, find the one with the smallest distance. Merge that pixel into that seed area if the pixel intensity is within  $\mu_k \pm 3\sigma_{R_k}$ .

The seeded-area-based region growing algorithm employs ordered lists of  $d(R_k, p_{i,j})$ . Fig. 2.5 shows a hypothetical example involving two seed regions (regions 1 and 2) and some unidentified pixels between them. The region-growing procedure is illustrated below. Table 2.1 lists the two sorted lists of the unidentified pixels adjacent to the two seed regions initially. Each row is sorted based on the distance of the unidentified adjacent pixels to the



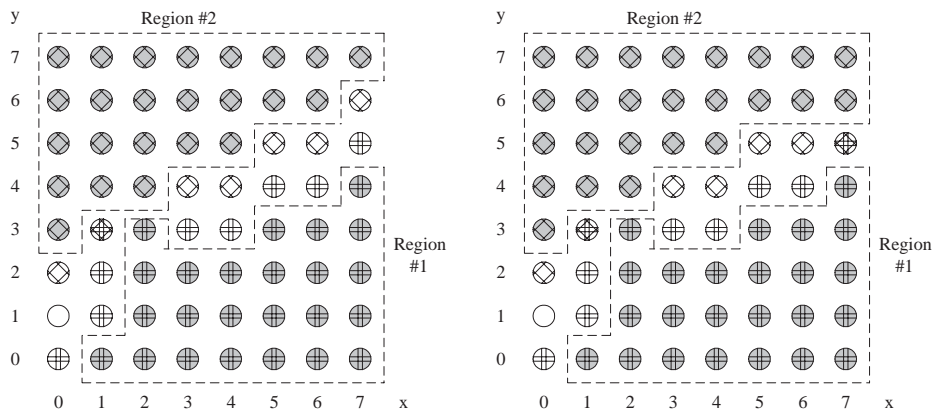
(a) Growing (3,2) into region #1

(b) Growing (2,2) into region #1



(c) Growing (2,3) into region #1

(d) Growing (7,7) into region #2



(e) Growing (6,6) into region #2

(f) Growing (7,6) into region #2

Fig. 2.6: Illustration of region growing.

Table 2.1: Sorted List of Adjacent Pixels for the Two Seed Regions

Candidates of Seed-region #	Sorted List of Adjacent Pixels							
	1	2	3	4	5	6	7	8
1	(3,2)	(2,2)	(4,3)	(0,0)	(1,1)	(5,4)	(6,4)	(7,5)
2	(7,7)	(6,6)	(0,2)	(1,3)	(3,4)	(2,3)	(4,4)	(5,5)

Table 2.2: Sorted List of Adjacent Pixels for the Two Seed Regions After Merging (3,2)

Candidates of Seed-region #	Sorted List of Adjacent Pixels							
	1	2	3	4	5	6	7	8
1	(2,2)	(3,3)	(4,3)	(0,0)	(1,1)	(5,4)	(6,4)	(7,5)
2	(7,7)	(6,6)	(0,2)	(1,3)	(3,4)	(2,3)	(4,4)	(5,5)

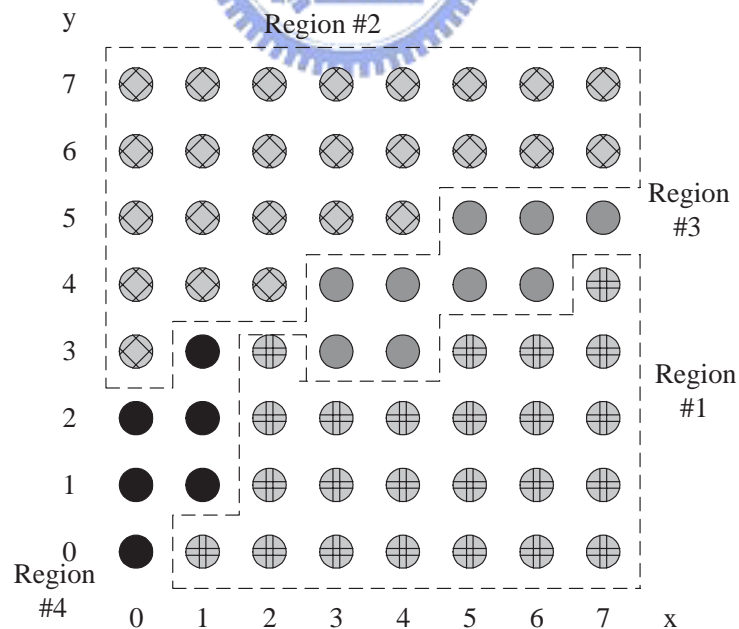
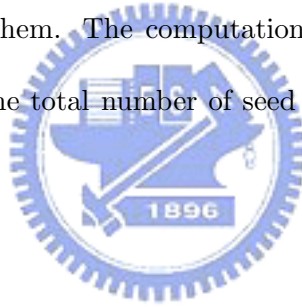


Fig. 2.7: Illustration of the final result of Fig. 2.5.

seed area  $R_k$ , namely,  $d(R_k, p_{i,j})$ . Furthermore, the priority in merging is given to the loading pixel with the lowest distance. Assume  $d(R_1, p_{3,2}) < d(R_2, p_{7,7})$ , so pixel (3,2) is merged into  $R_1$ . Fig. 2.6(a) illustrates the result after region 1 merges into  $p_{3,2}$  and the update of adjacent pixels.  $p_{3,3}$  then becomes adjacent to region 1 and the parameters of region 1 are recalculated, resulting in the sorted lists shown in Table 2.2. Similar operations are repeated until the seeded regions can grow no more. Figs. 2.6(a)–(f) illustrate the procedure assuming that (7,6) is the last growing pixel. Fig. 2.7 shows the final regions.

The complexity depends on the number of seed regions and their sizes. In the beginnings, we need to sort the adjacent pixels to each  $R_k$  according to their distances. In each step of region growing afterwards, we only need to re-calculate the distances for the pixels neighboring to the merged pixel and sort them. The computation of region growing has  $O(CN \log N)$  complexity, where  $C$  denotes the total number of seed regions in the process, and  $N$  is the number of pixels.



### 2.3.2 Region Merging

Watershed-type segmentation frequently leads to over-segmentation and causes the segmentation to lose the meaning of object identification. Region merging can reduce the amount of regions by fusing adjacent regions with high similarity.

In the proposed method, the measure of spatial similarity uses the regional mean. For each region  $R_j$ , its adjacent regions  $R_i$  are considered, and their spatial similarity is denoted by  $W_{i,j} = |\mu_{R_i} - \mu_{R_j}|$ . The  $W_{i,j}$  is sorted in a non-decreasing order, except for  $|R_i| > N_r$  and  $|R_j| > N_r$ . If  $|R_i| \leq |R_j|$ , then  $|R_i|$  is merged with  $R_j$ . The parameters (region mean and weight) of all other nodes connecting to  $R_i$  or  $R_j$  then are changed,  $W_{i,j} = \infty$  is set, and the weights are re-sorted. The cases  $(|R_i| + |R_j|) > N_r$  and  $(|R_i| + |R_j|) \leq N_r$  must be



considered further. Some examples are illustrated below. The procedure is iterated until no regions is smaller than  $N_r$ .

Fig. 2.8(a) is an arbitrary example for a region segmentation map. A RAG (Region Adjacent Graphic) is an undirected graph to represent the nodes corresponding to connected regions [54]. The links represent the vicinity relation and we weighted by similarity. Fig. 2.8(b) shows the corresponding RAG. Fig. 2.8(c) is assumed to be the ideal segmentation (simple two regions). The merging process then re-organizes the RAG in Fig. 2.8(d) into the fitness to Fig. 2.8(a). Initially, all  $W_{i,j}$  are shut away when both  $|R_i|$  and  $|R_j|$  exceed  $N_r$ . We assume  $|R_1| > N_r$  and  $|R_2| > N_r$ . So nodes 1 and 3 are marked with boldface circles, and we set aside the weight between nodes 1 and 3, i.e.,  $W_{1,3} = \infty$ . Figs. 2.9(a)–(d) merge the other thin-circle-nodes as follows:  $W_{4,6} = 1.0$  is the lowest value, and the region size with  $R_4 \geq R_6$  (see Fig. 2.8(a)), so  $|R_6|$  merges with  $|R_4|$ . Fig. 2.9(a) shows that the edge between nodes 4 and 6 becomes a dashed line (namely, edge-fusion), and all the other weights that connect with nodes 4 or 6 are re-computed. If  $|R_4 + R_6| \leq N_r$ , thus any node 4 and 6 remain merger candidates. Then as displayed in Fig. 2.8(d),  $W_{2,3} = 2.0$  is the smallest weight so that  $|R_2|$  merges with  $|R_3|$  (because  $|R_2| \leq |R_3|$ ). Fig. 2.9(b) shows that the circle of node 2 (because  $|R_3| > N_r$  and both nodes 2 and 8 are black circles (block  $W_{2,8}$ )), and re-calculate the weights of node-2-connected and node-3-connected nodes. The RAG illustrates that  $W_{3,4} = 3.1$  is the leading weight, so that  $|R_4|$  merges with  $|R_3|$  (because  $|R_4| \leq |R_3|$ ). Fig. 2.9(c) illustrates that circles of node-4 and node-6 are blocked (because  $|R_3| > N_r$  and nodes 4 and 6 are in a flock), and  $W_{1,4}$  and  $W_{2,4}$  are set aside (because both nodes 2, 4 and 8 are black circles). Then, only  $W_{1,5}$  remains, and can be fused as mentioned above motioned and illustrated in Fig. 2.9(d).

After sweeping up entire thin-circle-node. The question arises of what should be the next

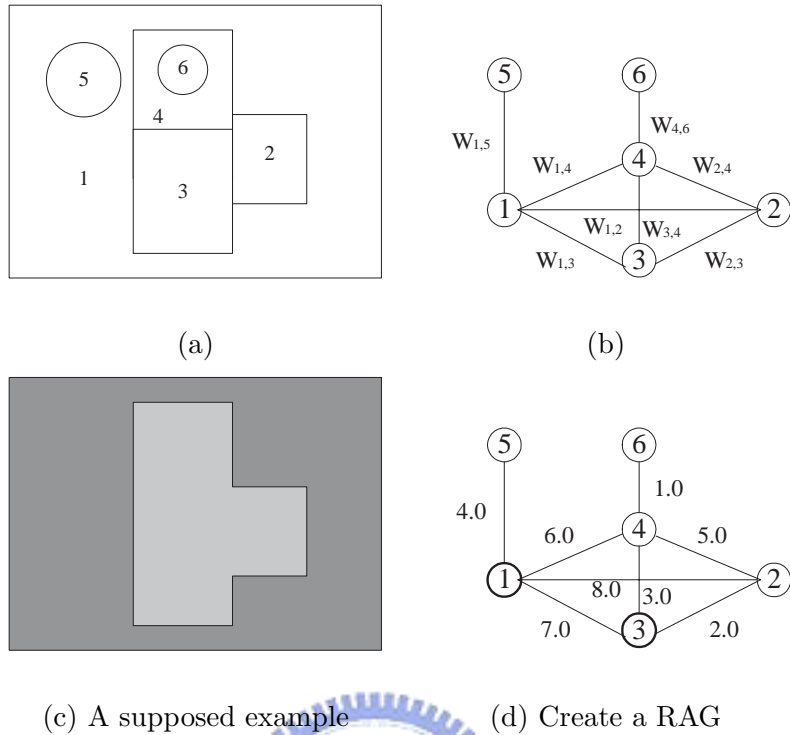


Fig. 2.8: Arbitrary example for illustration of the region-map and its corresponding RAG. (a) An arbitrary region-map example. (b) Corresponding weighted RAG. (a) Assume regions in (a), in ideal, become two regions after refining out small regions. (b) Weighted RAG.

step after sweeping up all of the thin-circle node. Fig. 2.9(f) finally shows that the dash-edge in RAG links with the merged regions, Exact two regions are left.

The region merging procedure is described as follows:

1. Given a K-RAG assume  $k$  regions are qualified for merger.
2. Sort weights.
3. Consider the minimum weight (consider  $i, j$  in turn and  $|R_i| \leq |R_j|$ ):
  - (a) Merge  $R_i$  and  $R_j$  and update the RAG parameters.
  - (b) Change the all weights related to  $R_i$  and  $R_j$ .

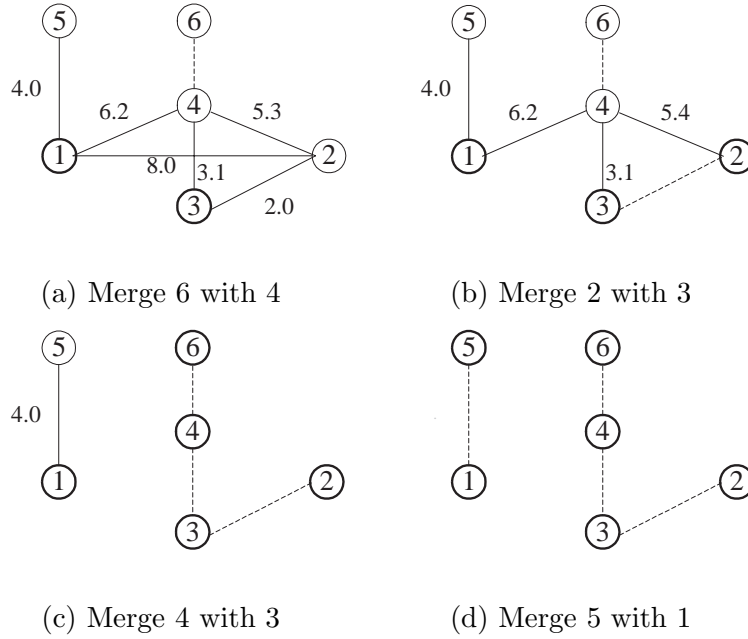


Fig. 2.9: Illustration of region merging and RAG changing: Merging Step-by-step.

(c) Re-sort locally.



4. Go to 3 if any  $|R_i|$  merge into  $|R_j|$ .

Next the complexity of the most expensive steps is examined. The weight sorting in step 2 costs  $O(|E|\log|E|)$  operations, where  $|E| = k \times (K - 1)$ . When each merging process, the complexity of 3a depends on regions' sizes. We can only say that this step takes about  $O(kN_r)$ . Step 3b fuses at least one weight, however, some weights must be removed. The illustration is shown in Fig. 2.9(d), where the node 2 merges with 3, and then we could black node-2 or block node-2's connections. The procedure takes  $O(K)$  for the worse case. Last, step 3c sorts the rest weights. At this time, the current weights are less than  $k$ , for convenience, we asses its complexity  $O(k\log|E|)$ . The complexity is thus approximately  $O(|E|\log|E|)$ .

Table 2.3: Sorting the RAG’s Weights

Rank	1	2	3	4	5	6	7
	$W_{4,6}$	$W_{2,3}$	$W_{3,4}$	$W_{1,5}$	$W_{2,4}$	$W_{1,4}$	$W_{1,2}$
Weights	1.0	2.0	3.0	4.0	5.0	6.0	8.0
$\Delta W$	-1.0	-1.0	-1.0	-1.0	-1.0	-2.0	-

## 2.4 Motion-Based Region Integration

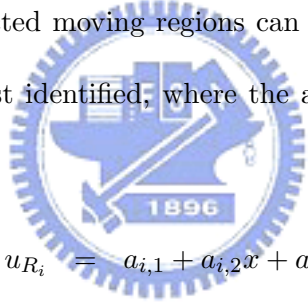
This task aims to estimate the motion of each region and extract the moving regions. To decompose a dynamic scene into its constituent objects, the temporal information frequently is important. Moving objects are mainly based on the temporal coherence. The assumption is that regions in the same object undergoes the same motion. Furthermore, it implies that the regions integrate into an object when they share high-similarity of motion. Thus, the temporal information is commonly used to determine whether two regions belong to the same object.

Briefly, techniques for defining the motion similarity can be divided into two types. One type derives directly from motion vectors [55, 56, 57, 58], while the other one is based on background statistics [1, 25, 26]. The former technique checks the distances between the parametric motion representations for two regions. Wang and Adelson [57] demonstrated that clustering the distance on a set of motion layers is practical. However, the inaccuracies of motion estimation may affect the segmentation results. Tuncel and Onural [55] estimated the dense motion vectors and proposed video-object segmentation by a fast recursive shortest spanning tree method. Moreover, Wang [58] integrated adjacent regions by measuring the incremental mean-square motion compensation error. The latter technique tested the

foreground regions based on the assumption of a still background.

The proposed method most closely resembles the first method. The method first constructs a dense motion field for each region using a forward motion estimation approach. To reduce the computational load while obtaining reasonably accurate motion vectors, this study first performs motion estimation for each pixel on the region boundary. The motion estimation employs block-matching techniques. The resulting motion vectors are called seed motion vectors. Each pixel in a region is tested only using these seed motion vectors. Regions with a high percentage of moving pixels are considered moving regions. Connected moving regions are integrated into a single big moving region provided certain criteria are met. Consistent with MPEG-4 terminology, each such integrated region may be called a VO.

To test whether two connected moving regions can be integrated, the affine motion parameters of each region are first identified, where the affine motion model for pixel  $p_{x,y}$  in region  $R_i$  is given by



$$\begin{aligned} u_{R_i} &= a_{i,1} + a_{i,2}x + a_{i,3}y, \\ v_{R_i} &= a_{i,4} + a_{i,5}x + a_{i,6}y, \end{aligned} \quad (2.2)$$

with  $u_{R_i}(x, y)$  denoting horizontal displacement and  $v_{R_i}(x, y)$  vertical displacement. The affine motion parameters  $a_{i,k}, k = 1, 2, \dots, 6$ , may be found with any appropriate error-minimization method. For convenience, let  $(u_{R_i}^{R_j}(x, y), v_{R_i}^{R_j}(x, y))$  denote the *synthesized motion vector* at pixel  $p_{x,y}$  in region  $R_i$ , obtained by substituting the affine motion parameters for region  $R_j$  into the (2.2) for region  $R_i$ . Then we do the following. For each pair of adjacent regions, say  $R_i$  and  $R_j$ , compute the sum of root-mean-square (RMS) motion vector errors with synthesized motion vectors as

$$C(R_i, R_j) = \sqrt{\frac{1}{N_{R_i}} \sum_{(x,y) \in R_i} [u_{R_i}^{R_i}(x, y) - u_{R_i}^{R_j}(x, y)]^2 + [v_{R_i}^{R_i}(x, y) - v_{R_i}^{R_j}(x, y)]^2}$$

$$+ \sqrt{\frac{1}{N_{R_i}} \sum_{(x,y) \in R_j} [u_{R_j}^{R_i}(x,y) - u_{R_j}^{R_i}(x,y)]^2 + [v_{R_j}^{R_i}(x,y) - v_{R_j}^{R_i}(x,y)]^2} \quad (2.3)$$

where  $N_{R_k}$  denotes the number of pixels in region  $R_k$ . Find the pair  $(i^*, j^*)$  with lowest error, that is,

$$(i^*, j^*) = \arg \min_{i,j} C(R_i, R_j) \quad (2.4)$$

If both the RMS errors in (Eqn.2.3) are below a predefined value, the two regions are integrated into one. The better set of affine motion parameters is used for the integrated region, and the same procedure is iterated until no more region integrations can be made.

## 2.5 Region Tracking and Updating

In image sequence segmentation for video compression and content-based functionalities, proper object tracking over video frames is extremely important, including the proper adjustment of object shapes over time to deal with possible object deformation, object occlusions from motion, and segmentation errors in earlier frames. By monitoring the progression of each object over the frames, a human viewer can, for example, select some objects of interest and observe their life over the video sequence.

For the task of region tracking and updating, each moving region is first projected into the next video frame using the forward motion information obtained via the motion-based-segmentation task. Some covered and uncovered areas may appear. Seed areas (found in the seed-area-identification task) are sought which have nonempty intersections with the projection footprint. For each such seed area, if more than a certain percentage of its size (for example, 50%) is in the footprint and the area of intersection is not below a predefined threshold (for example, 20 pixels), then the intersection is considered a valid new seed area. Additionally, pixels must have low percentage forecasting errors (for example, under 0.05) to

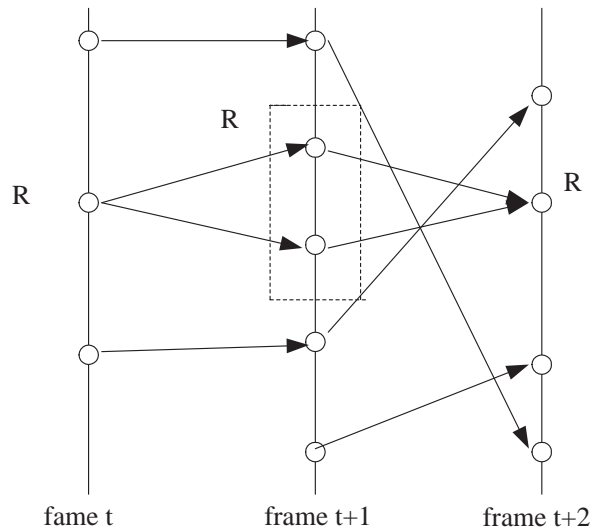
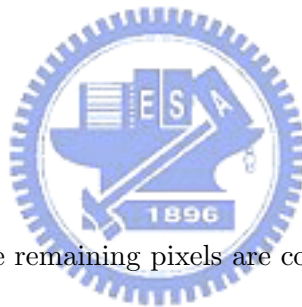


Fig. 2.10: Illustrative temporal progression of different regions.



be included certain ranges. The remaining pixels are considered “uncertain areas” requiring re-segmentation. For the uncertain areas, re-segmentation employs regional growth from the modified seed areas and region merging, as in the initial segmentation task.

The above procedure allows splitting, merging, and deformation of regions, as are needed. In tracking, one also needs to maintain the correspondence of regions in two successive frames. To deal with splitting and merging of regions, the newly segmented regions in the second frame which fall in the footprint of a region in the previous frame may be considered a valid child of the region. The idea is illustrated in Fig. 2.10 with the point marked R. The correspondence between regions in two successive frames are then affirmed, completing the work of region tracking and updating.

## 2.6 Experimental Results

To illustrate the performance of the proposed system structure, we now give some example results from segmenting some common test sequences. Consider first the Table Tennis sequence, where the picture resolution used is SIF ( $240 \times 352$ ). Fig. 2.11(a) is the original frame 1. Fig. 2.11(b) presents the region set of initial classification. In Fig. 2.11(c), we stand for the seed areas by white points and the black points are uncertain pixels. Fig. 2.11(d) shows the result of initial segmentation of the first frame. Fig. 2.11(e) shows the background portion of the second frame in the sequence. The dark-filled areas mark the locations of two connected moving regions in the foreground. Not dark-filled is a small, third moving region in the lower-right corner of the picture which shows the left hand of the table tennis player. The uncovered areas found in the second frame are shown in dark in Fig. 2.11(f).

Selected objects of Table Tennis from the segmentation results are shown in Figs. 2.12 and 2.13. The sampled result of the table tennis ball, tracked from frame 1 to frame 40, is shown in Fig. 2.12. The segmentation result every fifth frame of the ball-playing arm is shown in Fig. 2.13. The region tracking method gets the arm stationary linked across the frames. Fig. 2.14 shows the recreated frame 1 after we rebuild the Table Tennis's background, and removed the ping-pong ball and player's left hand.

Fig. 2.15 shows some tracking result of the foreground object in the CIF format ( $352 \times 288$ ) of Claire sequence. Note that the initial segmentation at frame 1 does not yield the full outline of the talker but only her head and parts of body. This is not surprising since we did not endow high-level intelligence into the segmentation algorithm. Considering the low-level processing, in the beginning of this scene, we can observe that the start motion is slight. As time progresses, the motion information is picked up and the segmentation becomes more in line with human perception.



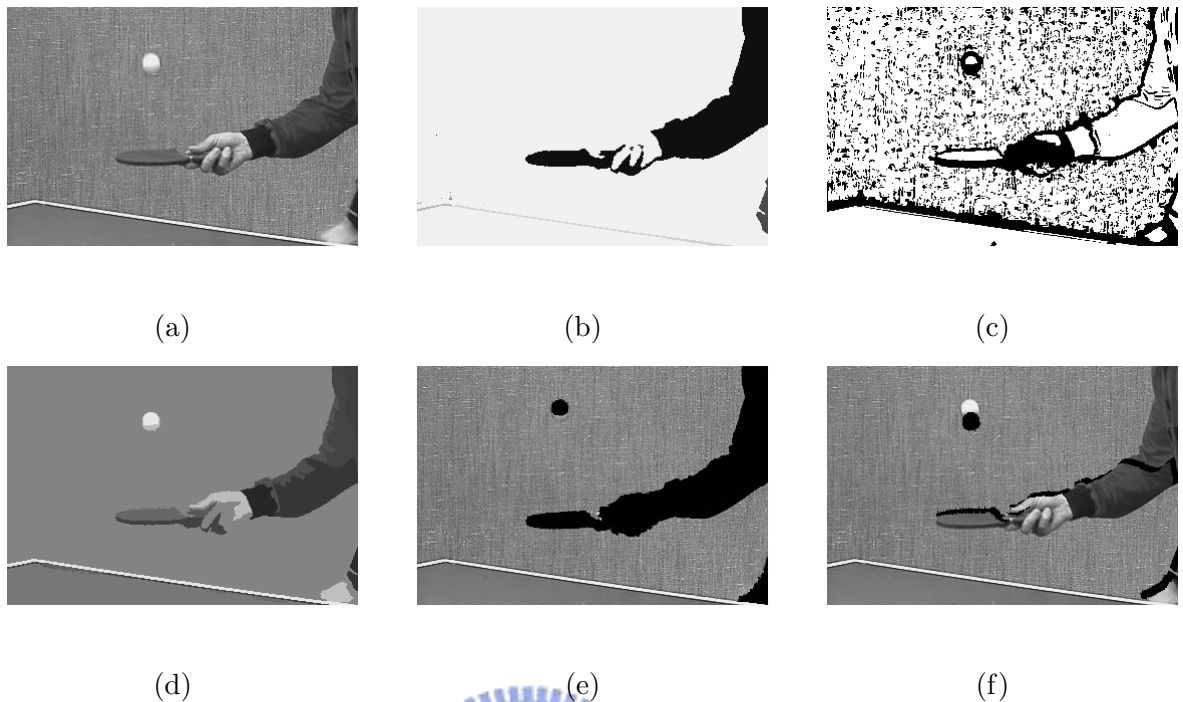


Fig. 2.11: Segmentation performance of the Table Tennis Sequence. (a) Original frame 1. (b) Regions by bright and dark segmentation. (c) Seed (bright areas) and uncertain regions (dark areas). (d) Initial segmentation result of the (a). (e) Background portion found in the second frame. (f) Uncovered area in frame 2 (dark points).

Fig. 2.16 shows sampled tracking result of the foreground object in the Akiyo sequence (CIF format). Here the outline of the talker appears better identified in frame 1. However, the upper-right part of the talker's head suffers continued redundant-segmentation due to likeness in intensity between the hair and the background. The right hair caused some mis-segmentation at the beginning, the problem disappeared in frame 20. However the later frames (showing of frame 80 and 100) occurred again. These examples confirm the effectiveness of the proposed method to some extent. They also indicate some areas for improvement.

Both Figs. 2.17 and 2.18 present the object performance. The curve in Fig. 2.17 shows the fractional agreement between our segmentation result and the reference mask for each



Fig. 2.12: Tracking of ping-pong ball.

frame evaluated as follows [60]:

$$Q^s(O_n^{seg}, O_n^{ref}) = 1 - \frac{\sum_{x,y} O_n^{seg} \oplus O_n^{ref}}{\sum_{x,y} O_n^{ref}} \quad (2.5)$$

where  $O_n^{seg}$  and  $O_n^{ref}$  are the segmentation mask and reference mask at frame  $n$ , respectively, and  $\oplus$  denotes the Exclusive-OR (XOR) operation. The  $Q^s$  is equal to or less than 1. The closer the value to 1, the better the result. Most of the accuracy values are above 0.98. The temporal coherency is defined by

$$\zeta_n = Q^s(O_n, O_{n-1}) \quad (2.6)$$

where  $O_n$  and  $O_{n-1}$  denote segmentation mask,  $O_n^{seg}$ , or reference mask,  $O_n^{ref}$ , at frame  $n$  and  $n - 1$ . Evaluating temporal quality measure is to test the following between  $O_n^{seg}$  and  $O_n^{ref}$ . Fig. 2.17 shows the spallity comparability of the extracted object to the mask in Akiyo sequence. Fig. 2.18 demonstrates the differences of each object's adjacency tightly followed the standard mask. We present some computing time data for CIF Akiyo, CIF Claire and SIF Table-Tennis. Tab. 2.4 lists the data from using a personal computer with 1.8-GHz Pentium IV CPU.

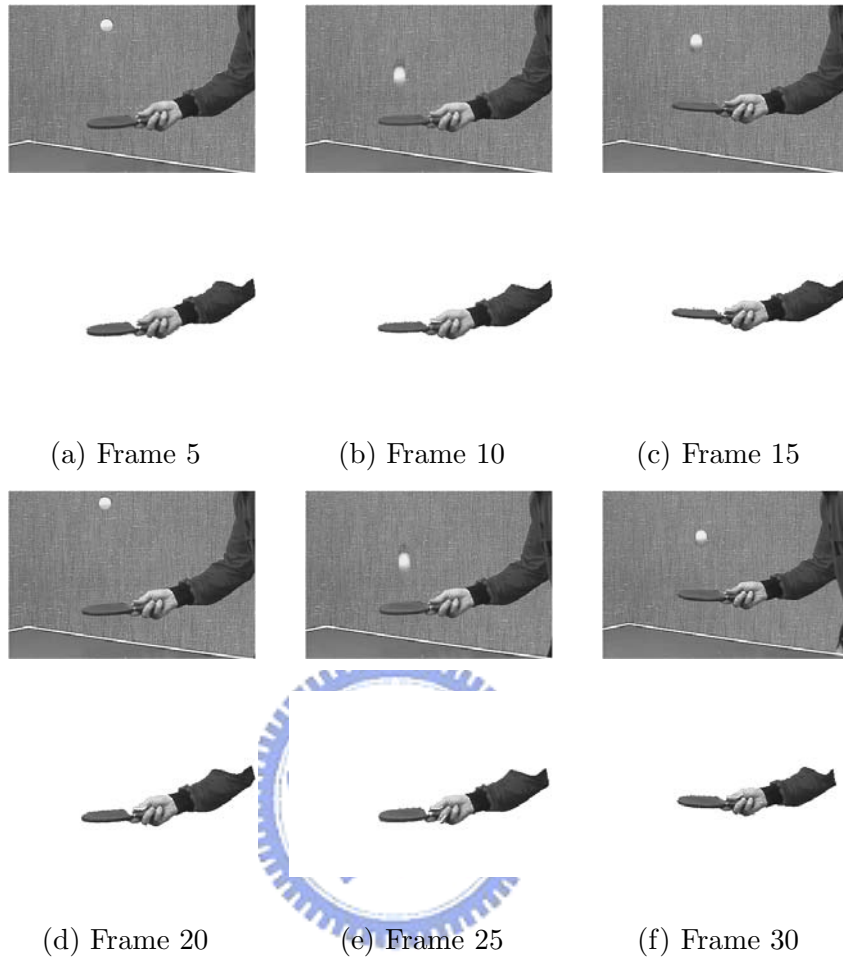


Fig. 2.13: Ball-playing-arm tracking result.

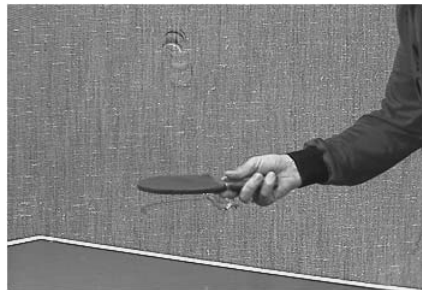


Fig. 2.14: Recreated frame 1 after removing ping-pong ball and player's left hand.

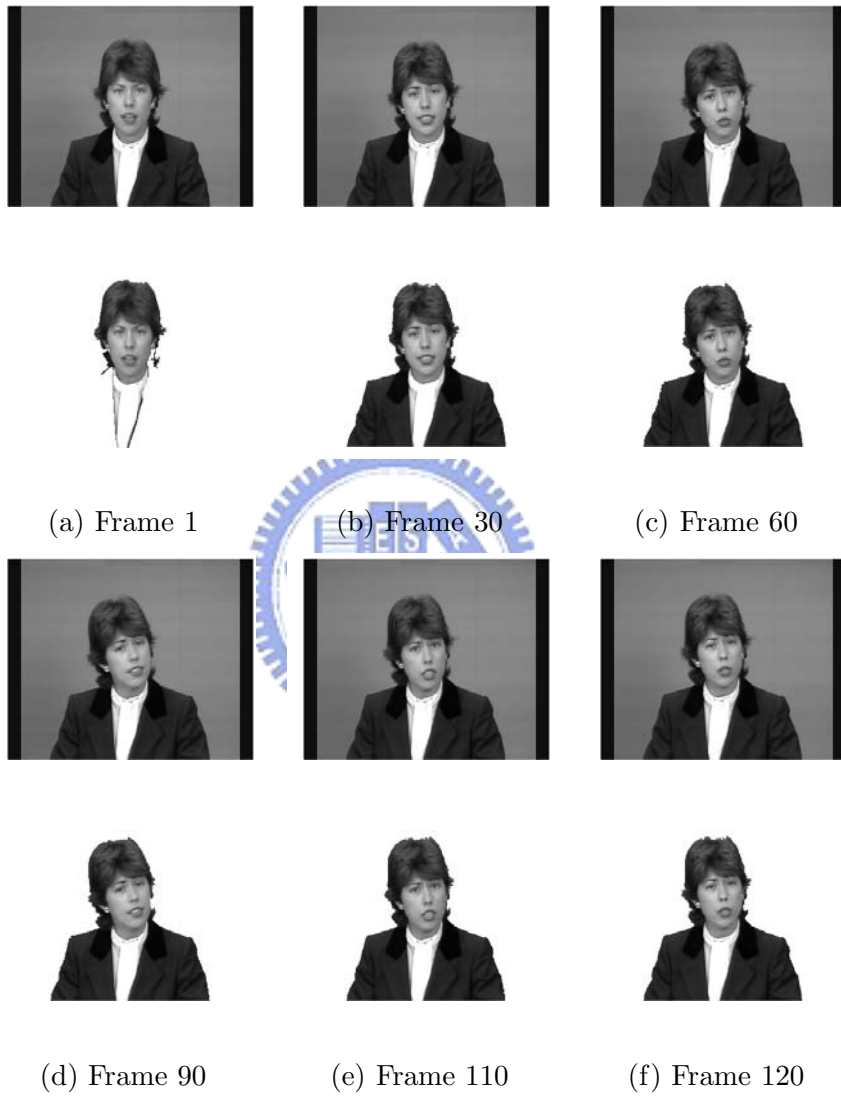


Fig. 2.15: VO tracking results of Claire.

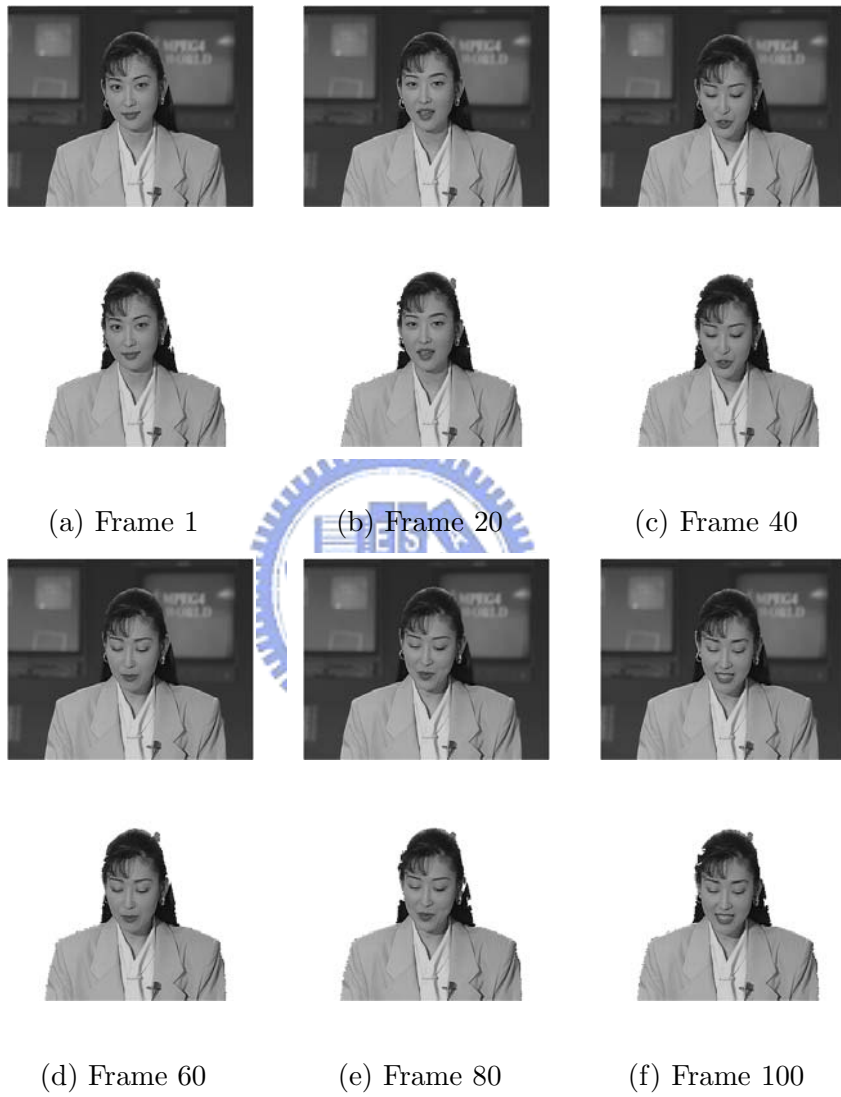


Fig. 2.16: VO tracking results of Akiyo.

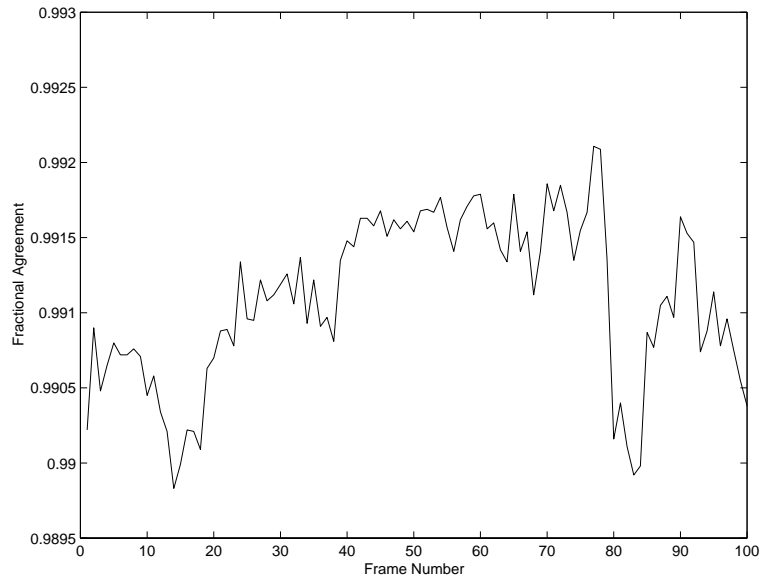


Fig. 2.17: Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask.

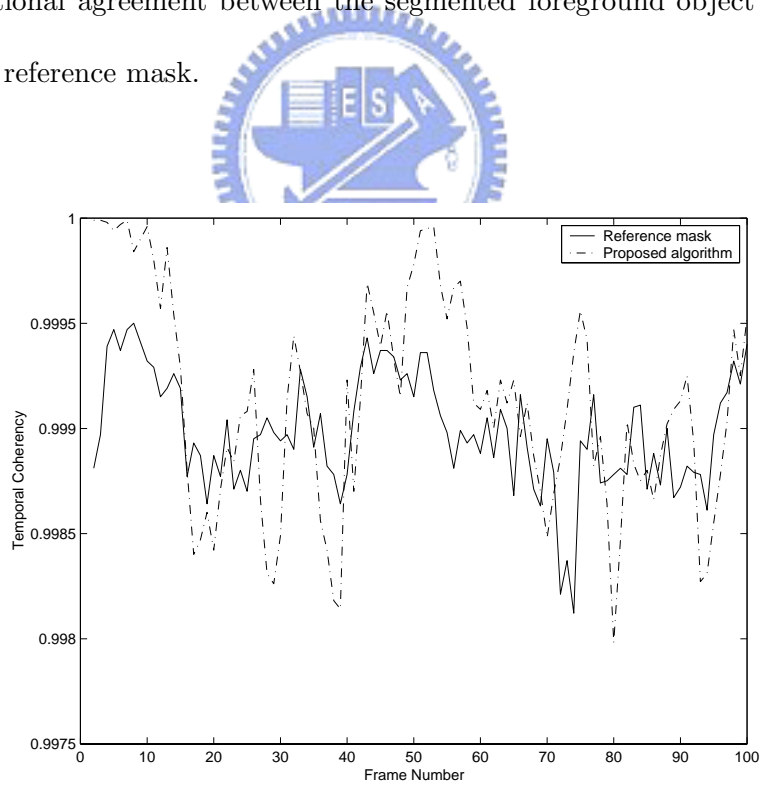


Fig. 2.18: Fractional agreement between the segmented foreground object masks of the Akiyo sequence in adjacent frames.

Table 2.4: Average Computing Time in ms per Frame of Major Algorithm Functions

Algorithm	Seed-Area	Region	Region	Motion	Total
Functions	Identification	Growing	Merging	Analysis	Time
Akiyo	33.1	50.1	13.1	223.5	319.8
Claire	37.5	53.3	14.8	230.9	336.5
Table-Tennis	39.6	68.7	29.6	339.7	477.6

## Chapter 3

# An Enhanced Region-Based Video Segmentation Algorithm with Extraction of Overlaid Objects



In the last chapter, we used region growing to split images into patches and used motion analysis to integrate VOs. This chapter presents an implementation for the case to extract the non-integral changed foreground and segment superimposed objects.\* In this chapter we propose a multi-tier processing which can extract the foreground containing multiple overlaid objects. We again employ low-level processing, i.e., texture and motion analysis. Some researches about image segmentation and object tracking have been described in Chap. 2 We begin by describing a brief review, point out some problems, and give a short introduction to the proposed multi-tier algorithm.

---

\*A major part of this chapter has appeared in Yih-Haw Jan and D. W. Lin, "Video segmentation with extraction of overlaid objects Via Multi-Tier Spatio-Temporal Analysis," *Int. J. Electrical Engineering*, vol. 11, no. 3, pp. 205–218, Aug. 2004.



### 3.1 Brief Review and Method Introduction

Before introducing the proposed algorithm, let us first briefly comment on some existing methods for fully automatic video segmentation. It is the experience of many that, in video segmentation, edges are important in accurate identification of object boundaries. Therefore, the watershed approach (which is essentially a sort of gradient or edge analysis) has been used frequently for spatial segmentation. A critical issue of the approach is how to threshold the gradient values to find proper region boundaries. Too low a threshold may result in gross over-segmentation (especially for highly textured scenes) and too high a threshold may combine multiple objects into one. The way to track the movement and shape change of each segmented region must also be designed carefully. Earlier results were apparently not satisfactory subjectively [58]. A later refinement yields significantly more accurate identification of region boundaries [17], but the way video objects are defined and tracked is rather simplistic, and there is no treatment of temporary overlaying together of multiple objects.

Taking a different route, Chen and Shirai [8] and Neri *et al.* [38] base their segmentation primarily on motion analysis. However, the issue of exact delineation of region boundaries is not addressed. In fact, the boundaries of the segmented regions can be quite far from actual object boundaries [38]. Since edges provide important cues to object boundaries, it is natural to expect a better segmentation from joint use of motion and edge information than consideration of motion alone. In fact, even the motion estimates are more reliable in edge and textured areas. In this vein, Gu *et al.* [19] conduct motion-based segmentation on edges. However, the segmented edges need not form closed contours and hence the results do not necessarily delineate each object fully. A few other algorithms employ motion estimation or change detection to locate moving regions and then use the edges found in the moving regions to define object boundaries [27, 36, 37]. Meier and Ngan [36, 37] consider separation of a

scene into two regions only, namely, the foreground and the background. However, the final algorithm [37] requires one to make an advance choice between use of motion estimation or change detection based on speed of object motion; where motion estimation is more suitable for slower motion while change detection, faster motion. If the motion is very fast, then background edges would have to be filtered out first. This is because the foreground region is determined by analyzing the edges in the changed regions (where the changed regions have been termed the change detection mask or CDM in short). If background edges are present in the CDM, then they may be mistaken to be part of the foreground and cause segmentation errors. Kim and Hwang [27] also employ change detection and edge analysis to extract the moving foreground. Then they conduct motion analysis to separate the foreground objects that exhibit different motion. However, the motion model is rather simple. Choi *et al.* [11] combine motion and edge analysis in an effective way, but the presented results do not lend themselves conveniently to evaluation of the segmentation accuracy at object boundaries. In addition, temporal tracking of the objects is not addressed. Kim *et al.* [26] rely on change detection and region analysis to segment the video. Since their object tracking does not use local motion information, we conjecture that the method may have difficulty dealing with fast moving, deformable objects.

We now introduce our algorithm. It has a tiered structure, in which the first (lowest) tier extracts a number of spatial and temporal features from the video and the upper tiers employ these features to effect the segmentation. In the design of the upper tiers, it is noted that, in video scenes, we often see moving objects (or parts of one bigger object) that show different motion overlaid one on top of another. Fig. 3.1 illustrates the situation schematically. There we depict the typical condition where there is a stationary background and a moving foreground. The foreground may contain several regions showing different motion, where

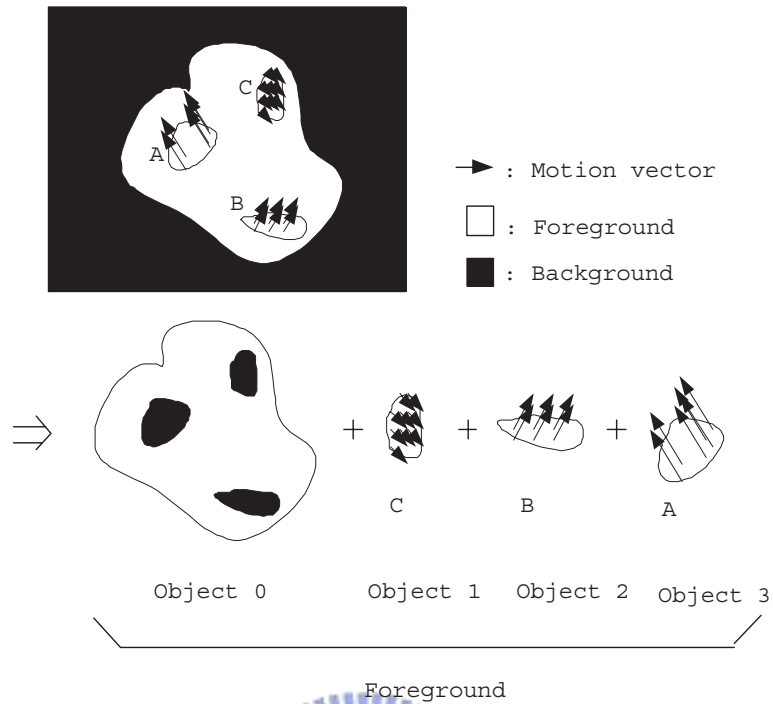


Fig. 3.1: Illustration of video scene with overlaid objects or region showing different motion.

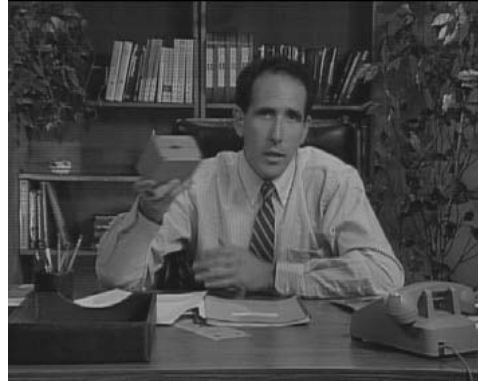


each region may actually be composed of one or more natural objects, or one or more distinct parts of a natural object. For the purpose of this work, these regions that are distinguished by difference in motion are called objects. Two real-life examples are given in Fig. 3.2. In the Flower Garden sequence, the foreground consists of the house, the flower bed, and the tree, with the tree showing different motion relative to the house and the flower bed. In the Salesman sequence, the person is in the foreground, with his arms showing different motion than the body. Our video segmentation algorithm can address this situation.

The algorithm is composed of three tiers and its overall structure is illustrated in Fig. 3.3. The low-level video features extracted by the first tier are edges, changed areas, block motion, and texturally homogeneous image regions (termed patches in this research). The second and middle tier employs these features in a motion-oriented analysis to effect a separation of the



(a) Flower Garden sequence



(b) Salesman sequence.

Fig. 3.2: Real-life examples where video scene contains multiple overlaid objects or regions showing different motion.

moving part (called foreground) and the stationary part (called background) in the video frames. The third and highest tier then identifies and tracks the overlaid objects, also via a motion-oriented analysis. Accordingly, the three tiers are named pixel tier, foreground tier, and overlays tier, respectively. As shown in Fig. 3.3, the pixel-tier functions can be further divided into two parts: edge analysis and motion analysis. Note that the designations of “foreground” and “background” as given above arise because a substantial fraction of video sequences is composed of a stationary background and moving foreground objects. The terms can be viewed as an artifice for convenience and need not be interpreted literally. In case where, due to camera motion, the physical background is moving but the physical foreground is stationary, the algorithm will consider the former foreground while the latter, background.

The remainder of this paper is organized as follows. Sec. 3.2 describes the motion analysis part in the pixel tier while Sec. 3.3, the edge analysis part. Sec. 3.4 describes the method of foreground separation. Section 3.5 describes the method of object segmentation and tracking for the foreground. Sec. 5.3 presents some experimental results. And Section 5.3 gives the

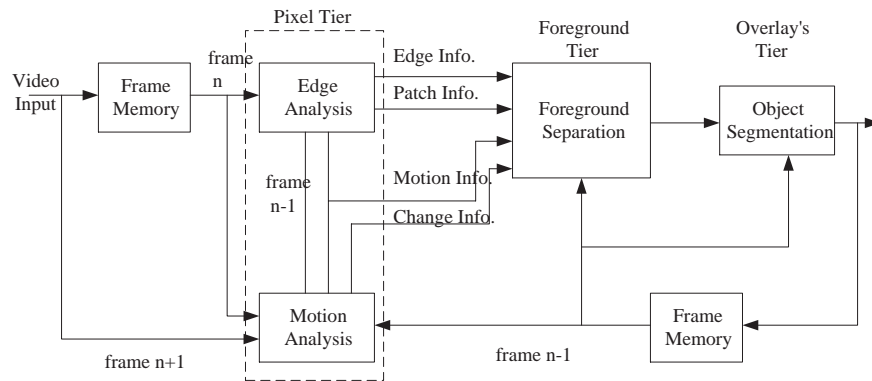
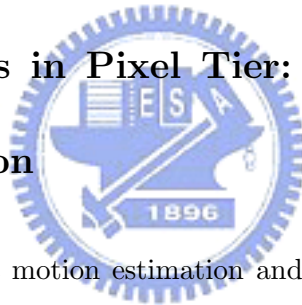


Fig. 3.3: Overall structure of the proposed algorithm.

discussion.

## 3.2 Motion Analysis in Pixel Tier: Motion Estimation and Change Detection



This part of the pixel tier does motion estimation and change detection to provide motion information and locations of changed pixels to the upper tiers. We describe the functions in separate subsections.

### 3.2.1 Motion Estimation

The motion estimation employs a block-based hierarchical technique [3] with the final block size set to  $2 \times 2$ . Use of the hierarchical technique with small final blocks results in relatively accurate motion estimates in reasonable complexity. The motion estimation is conducted in both the forward and the backward directions. That is, when we conduct motion estimation on two successive video frames, we divide both frames into  $2 \times 2$  blocks. For each block, the displacement with respect to the best-match block in the other frame is found.

The complexity of motion estimation depends on the parameters chosen for the hierarchical stages, such as the block sizes used, the search ranges, and the subsampling factors. The study has not made an extensive study concerning the optimal combination of parameters. Rather, we have made an arbitrary choice of a set that appears to yield acceptable performance. It results in an overall search range of  $\pm 14$  pixels in both the horizontal and the vertical dimensions. A more detailed discussion of the parameter settings would take us too far away from the focus of this work. Suffice it to say that the complexity for forward or backward motion estimation alone is roughly 97% of full-search block-matching, motion estimation with an equal search range. In fact, the motion estimation does not have to be carried out for every  $2 \times 2$  block in each direction. We shall see that only the foreground regions make use of the motion vectors. If the motion estimator avails itself of fee information concerning the location of the foreground regions, then the amount of computation incurred on motion estimation can be reduced considerably.

Conceptually, this corresponds to adding a feedback path from the foreground tier to the pixel tier in the algorithm structure. For example, if the foreground regions constitute 50% of the frame size, then the total complexity of our motion estimation scheme, forward and backward taken together, becomes roughly 97% of full-search block matchina ofeaul search range.

### 3.2.2 Detection of Changed Pixels

Change detection locates the moving video regions roughly. It requires statistical modeling of the background noise (due at least in part to camera noise). Typically, a pixel is declared changed (e.g., moving) or unchanged according to whether the frame difference around it is greater than or smaller than a threshold calculated from the statistical model. Because the

fundamentals of change detection have been well-documented in the literature [1, 25, 27, 38], we only describe our method briefly below. Let the video have stationary background. And assume the background part of the frame difference follows a zero-mean Gaussian distribution

$$p(d_k|H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d_k^2}{2\sigma^2}\right), \quad (3.1)$$

where  $H_0$  denotes the null hypothesis that the pixel at location  $(x, y)$  is unchanged,  $d_k$  is the difference at pixel  $(x, y)$  between frames  $k$ , and  $k+1$ , and  $\sigma^2$  is equal to twice the camera noise variance. Generally, We decide the threshold,  $TH$ , by required significance level. That is,

$$\alpha = p(d_k > TH|H_0), \quad (3.2)$$

where  $\alpha$  is the significance level. In this section, proposed method is geared at identifying the areas showing significant temporal variation in a video frame. To minimize the noise effects, the noise power in the frame is first estimated, so that object boundaries may be identified more robustly in the higher tiers. Model the observed frame  $n$  as

$$I_n = S_n + N_n, \quad (3.3)$$

where  $S_n$  is the signal and  $N_n$  is the noise, which may include camera noise and quantization error. Let  $\mu_s$  and  $\sigma_s^2$  denote the mean and the variance of  $S_n$ , respectively, and let  $N_n$  be zero-mean white Gaussian with variance  $\sigma_n^2$ . Further, assume that the signal and the noise are statistically independent. To estimate the noise power, we analyze the interframe intensity differences. The idea is a modified version of that in [?]. Let  $\delta_{m,n}$  denote the intensity difference between frames  $m$  and  $n$ , i.e.,  $\delta_{m,n} = I_m - I_n$  and let  $\delta_{m,n}(x, y)$  denote the value of  $\delta_{m,n}$  at pixel  $(x, y)$ . By the above model, the value of  $\delta_{m,n}$ , in the stationary background area of the video has Gaussian distribution. Note that

$$\sigma_{n-1,n}^2 = E[(I_{n-1} - I_n)^2] = 2\sigma_N^2 + 2\sigma_s^2(1 - \rho), \quad (3.4)$$

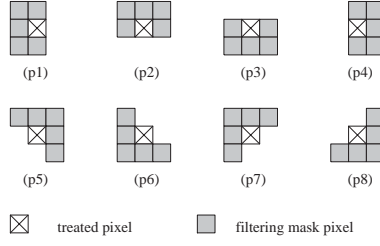


Fig. 3.4: Filtering masks for detecting moving pixels.

where  $\rho$  denotes the correlation coefficient between  $S_{n-1}$  and  $S_n$ . For stationary pixels, the intensity values are strongly correlated and  $\rho \approx 1$ . Therefore, the noise variance may be estimated through these pixels. For moving pixels, however, they may be nearly uncorrelated, i.e.,  $\rho \approx 0$ . These pixels should be excluded in the estimation of the noise power. We determine these pixels in the following way. Essentially, it finds the pixels around which  $\rho_{n-1,n}$  shows a directional structure, and considers them as moved. Consider the set of filtering masks shown in Fig. 3.4. For each pixel  $(x,y)$ , we compute the eight mean intensity differences defined by the eight masks  $P_i$ , as

$$f_i(x, y) = \frac{1}{5} \sum_{(s,t) \in P_i} \delta_{n-1,n}(x, y). \quad (3.5)$$

Let  $\mu(x, y)$  and  $\sigma_m(x, y)$  be the mean and the variance of  $\delta_{n-1,n}$  in the  $3 \times 3$  W centered at  $(x,y)$ , i.e.,

$$\mu(x, y) = \frac{1}{5} \sum_{(s,t) \in P_i} \delta_{n-1,n}(x, y), \quad (3.6)$$

$$\sigma_m^2(x, y) = \frac{1}{9} \sum_{(s,t) \in W} [\delta_{n-1,n}(s, t) - \mu(x, y)]^2. \quad (3.7)$$

And let  $\mu_{\delta_{n-1,n}}$  be the mean of  $\delta_{n-1,n}$ . Define

$$F(x, y) = \begin{cases} \delta_{n-1,n}, & \text{if } |f_i(x, y) - \mu_m(x, y)| < \alpha \sigma_m(x, y) \forall i, \\ \mu_{\delta_{n-1,n}}, & \text{otherwise,} \end{cases} \quad (3.8)$$



and  $\alpha$  is some suitable constant. Then we estimate the noise power as

$$\hat{\sigma}_N^2 = \frac{1}{2M} \sum_{(x,y)} [F(x,y) - \mu_{\delta_{n-1,n}}]^2, \quad (3.9)$$

where  $M$  is the number of pixels in a frame. The noise power estimate may be used to define a threshold  $\beta\hat{\sigma}_N^2$ , where  $\beta$  is some suitable factor, so that the pixels  $(x, y)$  in frame  $n$  for which  $\sigma_m^2(x, y) < \beta\hat{\sigma}_N^2$  are declared unchanged while others are declared changed. A similar process may be performed on  $\delta_{n,n+1}$  to determine another set of changed pixels. The intersection of the two sets is considered changed pixels in frame  $n$ . The reason for taking the intersection is as follows.

For a moving object, the changed pixels identified using two successive frames may include the areas around the object that are covered or uncovered due to object motion. Taking the above intersection helps localize the object better for the upper tiers' benefit. Fig. 3.5 illustrates the situation, where an object (or the boundary of a smooth object) denoted  $R$  is shown moving over frames  $n - 1$  to  $n + 1$ . Taking the intersection of two successive sets of changed pixels may reduce (or even eliminate) the covered and uncovered areas and better localize the moving object in frame  $n$ . Of course, if the object does not move between frames  $n - 1$  and  $n$  or between frames  $n$  and  $n + 1$ , then ideally the algorithm would return a null set for it. In addition, the changed pixels for the first frame of the video sequence are obtained from analyzing the first and the second frames only.

### 3.3 Edge Analysis in Pixel Tier: Edge Detection and Patch Delineation

This part of the pixel tier does detection of edges and partitioning of a video frame into texturally homogeneous image regions. This is to provide edge and region boundary information

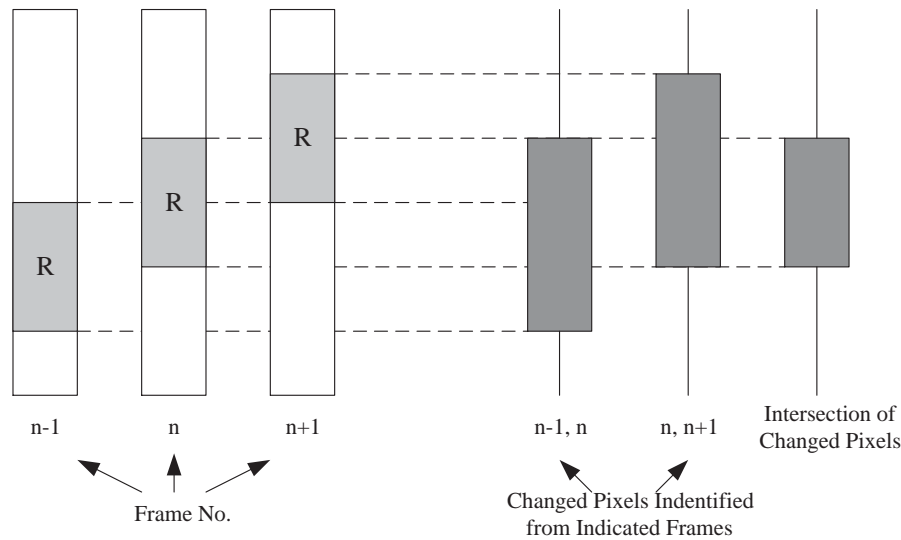


Fig. 3.5: Reducing covered and uncovered areas in change detection.

for upper tiers' use. For convenience and to avoid likely confusion with subsequent use of the terms "region" and "object," the texturally homogeneous regions so obtained will be termed "patches" and the process to obtain such regions "patch delineation." Edge information is very useful in detecting object motion and identifying object boundaries. We employ the Canny method for edge detection [5]. However, typical edge detection algorithms, including the Canny method, do not necessarily yield closed region contours or fully connected curves at object boundaries. To help object segmentation, therefore, we also conduct intensity-and-texture-based patch delineation for a complement. Experience shows that patch delineation based on simple intensity and texture analysis may encounter difficulty in highly textured areas and at thin objects (such as rims of picture frames and ends of bookshelf layers), where the patches fanned may not be aligned with natural texture or object boundaries. Edge detection may perform better in these cases. In conclusion, it is deemed that the joint use of edge detection and patch delineation should yield complementary information and thus be

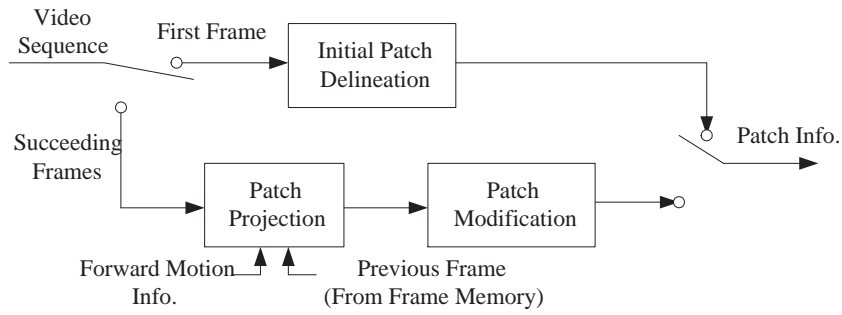


Fig. 3.6: Patch delineation algorithm.

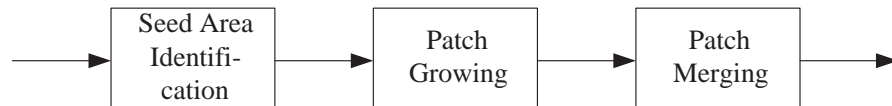


Fig. 3.7: Process of initial patch delineation.

beneficial.



As the Canny edge detection method is well-documented in the literature, below we only describe our method for patch delineation, which is a special feature of the overall segmentation algorithm. Its procedure is illustrated in Fig. 3.6. As shown, it consists of three functional blocks, namely, initial patch delineation, patch projection, and patch modification. We describe these blocks in the following subsections.

In principle, the delineated patches do not have to constitute semantically meaningful objects, but due to the structure of natural video scenes, they often will correspond to semantically meaningful parts of bigger objects.

The patch delineation was mentioned in Chapter 2 where the terms “region” and “patch” are a synonym. The task is accomplished in three steps: seed-area identification, patch growing, and patch merging. Fig. 3.7 depicts the process.

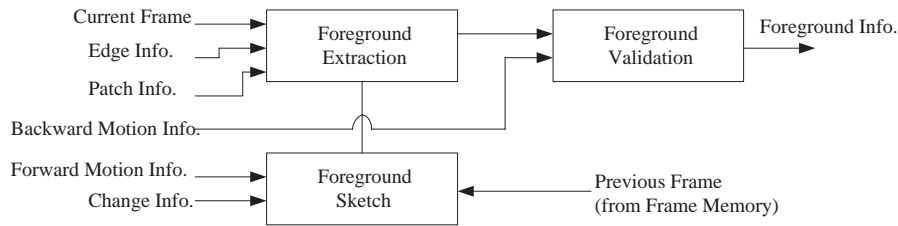


Fig. 3.8: Structure of the foreground separation algorithm.

### 3.4 Foreground Tier: Motion-Oriented Foreground Separation

This tier separates each video frame into two parts: a moving part called foreground and a stationary part called background. The foreground may consist of more than one connected set of pixels. For convenience, we term each such connected set a foreground region. The next tier will further divide each foreground region into objects based on further motion analysis.

The structure of our algorithm for foreground separation is shown in Fig. 3.8. The step of foreground sketch obtains a sketch of the foreground regions in the current frame by working with the changed pixels in the current frame and the footprints of the motion-compensated projection of the foreground regions found in the last frame. The step of foreground extraction refines the sketch by determining a more exact border for each foreground region (using the edge and patch information from the pixel tier) and then judiciously fills the space inside the border. Lastly, the step of foreground validation adjusts the boundary of each foreground region so extracted by projecting it backwards into the earlier frame and comparing its shape with the corresponding foreground region in the last frame. Large discrepancies may give birth to new foreground regions. The steps are described in detail in the following subsections.

### 3.4.1 Foreground Sketch

This step forms a sketch of the foreground regions for the current frame (say  $I_n$ ) for later refinement.

We first perform motion-compensated projection of the foreground pixels found in the last frame  $I_{n-1}$  into the current frame, employing the forward motion vectors obtained in the pixel tier. (This projection is omitted for the first frame of the video sequence.) We then include the changed pixels found in the pixel tier in constructing the sketch. This is to accommodate minor changes in object shapes that are not captured by block-based forward motion estimation and to allow possible addition of new moving objects with time. Specifically, we take the union of the projected foreground map (denoted  $F_n^p$ ) and the set of changed pixels (denoted  $C_n$ ). Mathematically, let  $\cup$  denote the union, then  $U_n = F_n^p \cup C_n$ .

For a large moving object with smooth intensity variations, it is possible that only pixels on the boundary are marked as changed. As a result, the changed areas identified in the pixel tier may have holes in the middle. This condition may hold even after taking the union with the projected foreground. To flesh out the interior of the foreground regions, we execute a fill-in process as follows.

First, each connected set of pixels in  $U_n$  is considered to define a foreground region, if its area exceeds a certain threshold, say  $A_r$ . For convenience, let  $U_n$  denote one such set. For each  $U_n$ , a “row fill-in” and a “column fill-in” are conducted, where the former fills in the space between the leftmost and the rightmost pixels of each row in the set and the latter fills in the space between the topmost and the bottom most pixels in the set. The union of the two resulting maps is taken. Then the union is eroded from the outer side so that the outer pixels that are not the eight-neighbor of any pixel in  $U_n$  are etched away. The result constitutes the desired foreground sketch.

### 3.4.2 Foreground Extraction

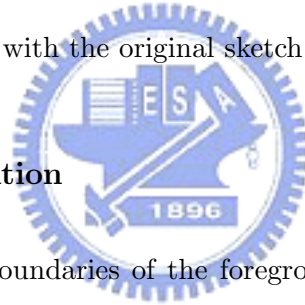
Built on the result of the foreground sketch and the edge and the patch information from the pixel tier, this step determines a more exact border for each foreground region and then does a judicious fill-in of the space inside the border. The refined border for each foreground region is obtained by finding, within the sketch of the region, among the edge pixels and the patch boundary pixels those that are closest to the sketch's perimeter. A row fill-in and a column fill-in, as that conducted in the step of foreground sketch, are performed over this set of border pixels. Instead of taking the union as in the step of foreground sketch, we now take the intersection of the resulting maps. To smooth out the border, a morphological dilation [18] is executed on the intersection map using a  $3 \times 3$  structuring element. Only the intersection of the dilated map with the original sketch of the region is kept.

### 3.4.3 Foreground Validation

This step further refines the boundaries of the foreground regions. It can also detect and separate new foreground regions that emerge next to the existing ones and hence may get extracted together with them in the two previous steps.

To fine-tune the foreground boundaries, we perform backward motion-compensated projection (from  $I_n$  to  $I_{n-1}$ ) for the foreground regions in  $I_n$ , employing the backward motion vectors found previously via hierarchical motion estimation. For an extracted foreground region in  $I_n$ , the pixels whose backward motion do not point them to the corresponding foreground region in  $I_{n-1}$  are trimmed from the region and considered as possibly belonging to a new foreground region.

This trimming may possibly result in some holes in a foreground region map and jaggedness in its boundary. Therefore, the region is subjected again to the fill-in process as that



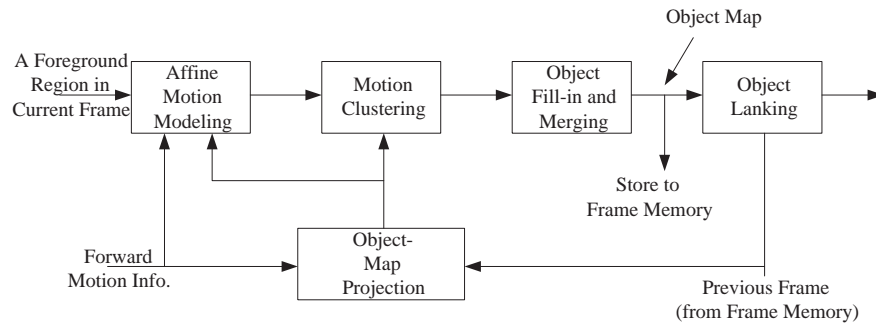


Fig. 3.9: Structure of object segmentation algorithm.

done in the steps of foreground sketch and foreground extraction. The set of trimmed pixels are also subjected to the same process, so that any new foreground region may be detected.

Now, to further distinguish the different semantic objects in each foreground region, we execute a motion-based segmentation process in the overlays tier as described next.

### 3.5 Overlays Tier: Segmentation of Overlaid Objects

This tier first segments the foreground regions that are separated in the last tier to obtain semantically meaningful objects. This is done through clustering of the foreground pixels by similarity in motion. A spatially connected cluster will potentially seed an object. Time links between the finally obtained objects (more exactly, video object planes in MPEG-4 language) in successive video frames are established via a particular procedure and the objects are tracked through the video sequence. Fig. 3.9 depicts the algorithm structure. The details are explained below.

The motion clustering is based on affine motion parameters, and we only use such parameters from the foreground areas that show homogeneity in motion. To identify the areas with homogeneous motion,  $I_{n-1}$  and  $I_n$  are first divided into fixed-size blocks. The block size

may or may not be equal to the final block size in the hierarchical forward motion estimation of the pixel tier. For each foreground region in  $I_{n-1}$ , blocks belonging to the interior of each segmented object in  $I_{n-1}$  are projected (with motion compensation) into  $I_n$ . (This projection is skipped if  $I_n$  is the first frame to be segmented.) Let  $P$  denote the union of the footprints of this projection. For each foreground region in  $I_n$ , say  $F$ , let  $S = F \cap P$ . Then the pixels of  $S$  in each block form one area for which we check for homogeneity in forward motion. Mathematically, let  $(u(x, y), v(x, y))$  be the motion vector at pixel  $(x, y)$ . Let  $B_i$  denote the  $i$ th block in  $I_n$ . We calculate the variance in motion of the pixels in  $S \cap B_i$  as

$$\sigma^2 = \frac{1}{N} \sum_{(x,y) \in S \cap B_i} [u(x, y) - \bar{u}]^2 + [v(x, y) - \bar{v}]^2, \quad (3.10)$$

where  $N$  is the number of pixels in  $S \cap B_i$ , and  $(\bar{u}, \bar{v})$  is the mean forward motion vector in  $S \cap B_i$ . If  $\sigma^2$  is greater than a predefined threshold  $T_d$ , then the motion of the foreground region  $F$  in  $B_i$  is considered inhomogeneous and skipped in the clustering operation. (Therefore, this check of motion homogeneity is not needed when the size of  $B_i$  is the same as the final block size used in the hierarchical forward motion estimation in the pixel tier, for in this case all pixels in  $B_i$  share the same motion vector.) For each retained  $B_i$ , the affine motion parameters  $a_{ik}$ ,  $k = 1, \dots, 6$ , for  $S \cap B_i$ , are obtained, where the affine motion model is defined similarly to (2.2). This completes the step marked “affine motion modeling” in Fig. 3.9.

In the step of motion clustering, the K-means clustering algorithm is applied to the set of affine motion parameters of the retained blocks to yield an appropriate number of motion classes, say  $N_c$ . Each pixel in foreground  $F$  is assigned to the class that matches best in motion. To speed up the convergence of the K-means algorithm, for the second and later frames it may be initialized with the help of the previous frame’s segmented object map. Specifically, the resulting object map for the previous frame may be projected (with motion compensation) into the current frame. The mean affine motion parameters of the retained



blocks in the projected footprint of each object may be used to initialize the clustering algorithm. We note that Wang and Adelson [57] also employ motion-based clustering to segment video scenes into layers, but the detailed procedure is quite different. After motion clustering, each connected set of pixels in a foreground region is considered to potentially define an object. Again, these sets may not form filled areas but may have holes in the middle. A fill-in process as that in foreground sketch is used to clear out the holes. Overly small objects are merged into a large adjacent object based on likeness of the mean affine motion parameters. The above constitutes the work in the step of “object fill-in and merging.” Finally, in the step of object linking, we establish temporal linkages of the segmented objects across video frames. The mechanism therein allows new appearance, disappearance, and merging of objects. In essence, it compares the footprints of the forward motion-compensated projection of the objects in the previous frame with the object map of the current frame to determine the temporal linkings and form tracking paths. The details are as follows. For an object in frame  $I_{n-1}$ , say  $O_{n-1}$ , let the forward motion-compensated projection in  $I_n$  be  $O_n^p$ . Assume that  $O_n^p$  intersects with  $q$  segmented objects in  $I_n$ , denoted  $O_n^m, m = 1, 2, \dots, q$ . Let  $A_m$  denote the area of  $O_n^m$  and let  $A_m^c$  be the area of the intersection between  $O_n^p$  and  $O_n^m$ . Any object  $O_n^m$  for which this intersection constitutes an over-whelming percentage of the object size, i.e.,  $A_m^c/A_m \geq \Omega$  for some large percentage number  $\Omega$ , is considered a successor of  $O_{n-1}$  in  $I_n$ . If no such  $O_n^m$  exists, then  $O_{n-1}$  ceases to exist. (For example, it may have merged into another object.) Objects in  $I_n$  that are not the successor of any object in  $I_{n-1}$  may start new tracking paths.

## 3.6 Experimental Results

We present some experimental results for the sequences Akiyo ( $352 \times 288$ ), Salesman ( $352 \times 288$ ), and Flower Garden ( $720 \times 480$ ). The first two have stationary backgrounds while the third has camera pan. Our experience indicates that the Salesman sequence is quite a challenge to video segmentation algorithms and we have not seen its use in the segmentation examples reported by others.

To start, we show some pixel-tier results. Since Canny edge detection and the hierarchical motion estimation are well-documented operations, we only present the results of patch delineation and change detection. Fig. 3.10 shows some patch delineation results and Fig. 3.11 shows some results of change detection. The results are subjectively proper.

Next, we consider the results of the foreground tier. Fig. 3.12–3.14 show the foreground yielded by the algorithm for the different sequences. Note that the foreground boundaries are identified reasonably well, except in the case of the Flower Garden where parts of the sky are grouped with some tree branches into the foreground. A solution to this last problem awaits further work. We now consider the results of the overlays tier. For the Akiyo sequence, since the foreground (the news woman) does not contain parts with significantly different motion, it is not segmented into multiple objects. In other words, the overlays and the foreground tiers have the same results. The Salesman and the Flower Garden sequences are different.

For Salesman, the arms and the head show different motion than the body. Figs. 3.15 and 3.16 show the right arm and the head as segmented in the overlays tier. They are tracked successfully in object linking. For Flower Garden, Figs. 3.17 and 3.18 show the objects “tree trunk” and “house right” as segmented from the foreground. Fig. 3.19 demonstrates the ability of the algorithm to deal with changes in object shapes; in this case it successfully tracks the changing shape of the building emerging from the right of the tree trunk.

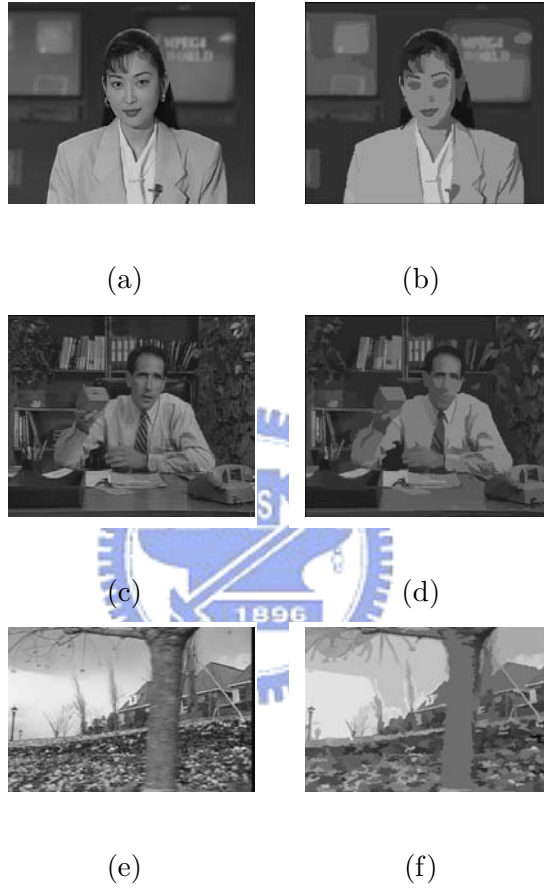


Fig. 3.10: Some patch delineation results for Akiyo, Salesman, and Flower Garden. (a),(c), and (e) are the originals, and (b), (d), and (f) are the results, where the different patches are distinguished by different gray levels.

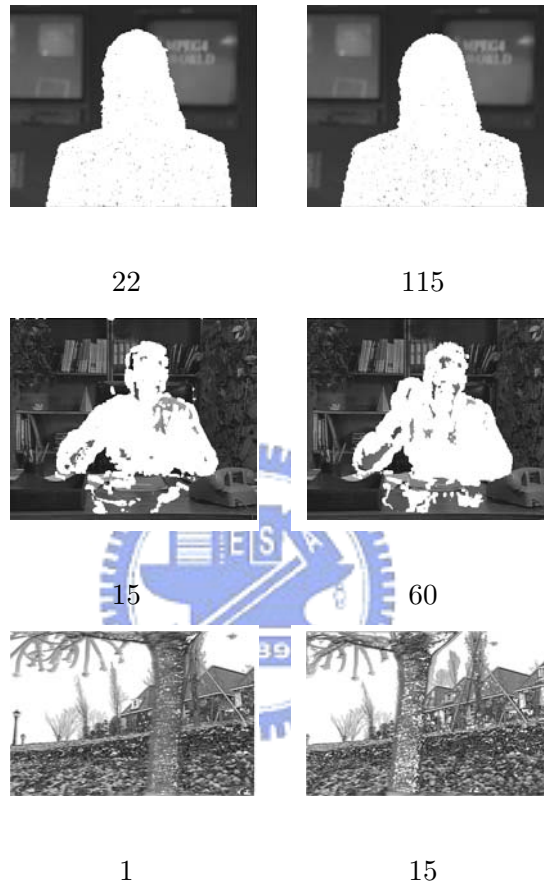


Fig. 3.11: Some change detection results for Akiyo, Salesman, and Flower Garden. Number below each picture gives the frame number. In the case of Akiyo and Salesman, the changed pixels are marked in white, but for clarity, in the case of Flower Garden the unchanged pixels are instead marked in white.

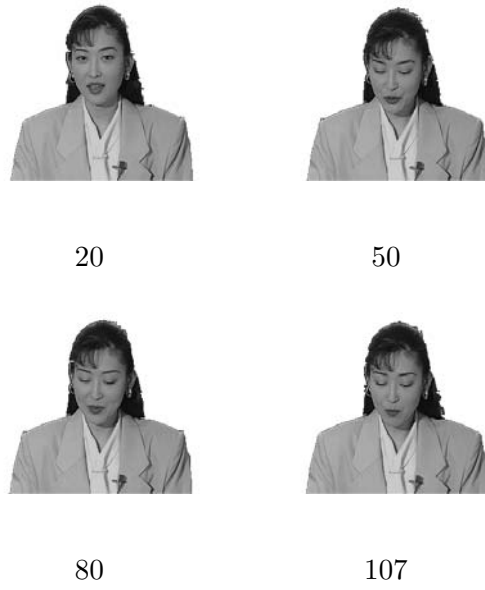


Fig. 3.12: Foreground region of Akiyo. Number below each picture gives the frame number.

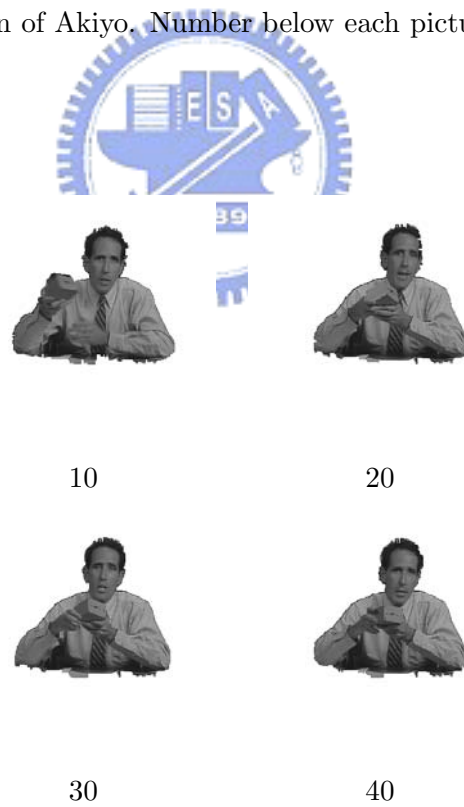


Fig. 3.13: Foreground region of Salesman. Number below each picture gives the frame number.

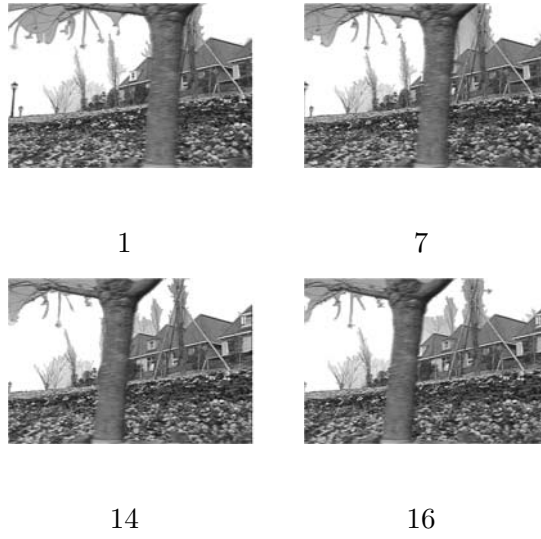


Fig. 3.14: Foreground region of Flower Garden. Number below each picture gives the frame number.

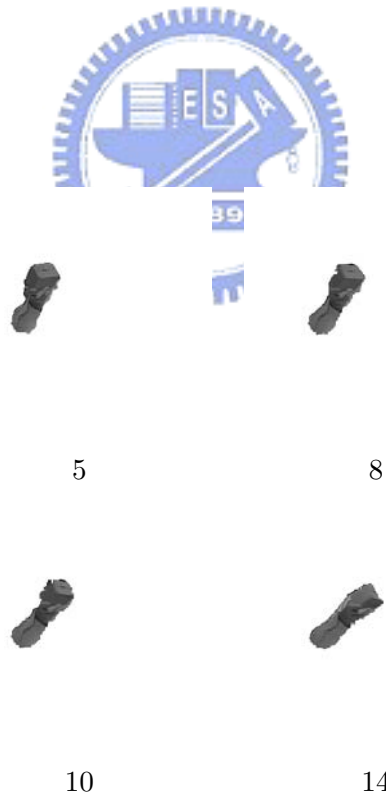


Fig. 3.15: Object “right arm” obtained in the overlays tier from the foreground region of Salesman. Number below each picture gives the frame number.

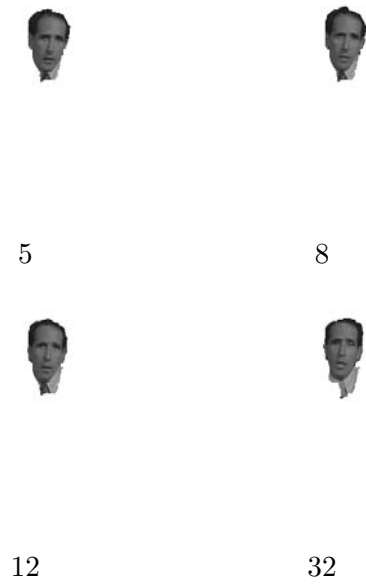


Fig. 3.16: Object “head” obtained in the overlays tier from the foreground region of Salesman. Number below each picture gives the frame number.

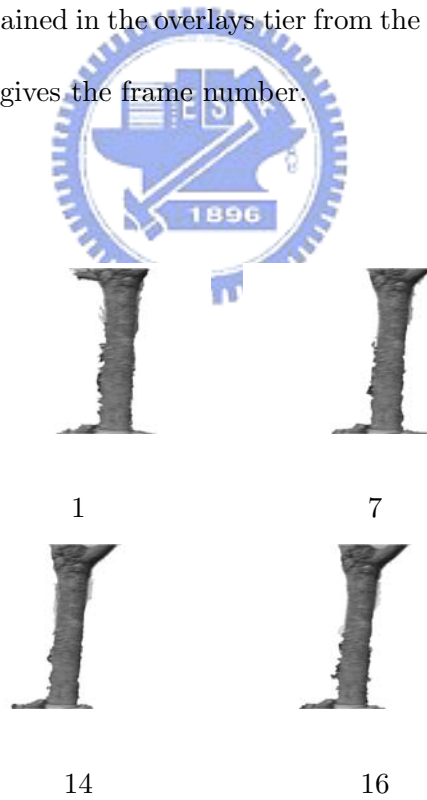


Fig. 3.17: Object “Tree trunk” obtained in the overlays tier from the foreground region of Flower Graden. Number below each picture gives the frame number.

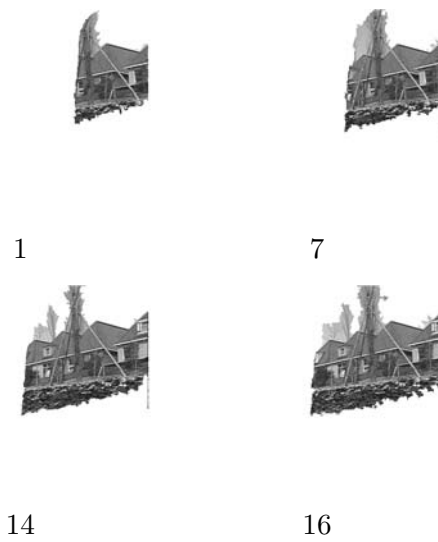
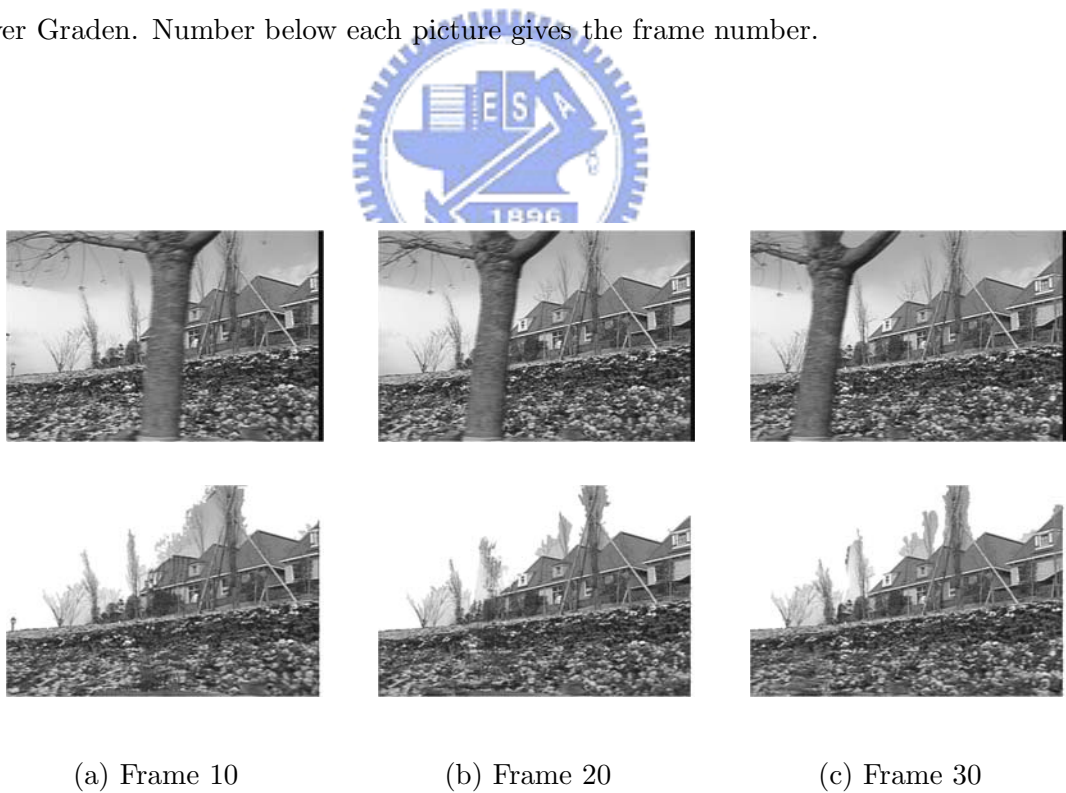


Fig. 3.18: Object “house right” obtained in the overlays tier from the foreground region of Flower Graden. Number below each picture gives the frame number.



(a) Frame 10

(b) Frame 20

(c) Frame 30

Fig. 3.19: Synthesized Flower Garden scene from segmentation result, with tree removed.

Upper row: original frames 10, 20, 30; middle row: synthesized frames.



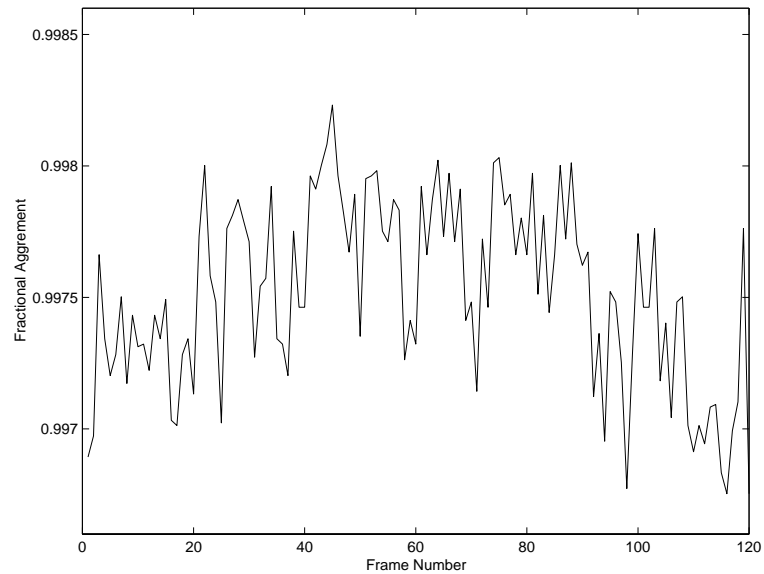
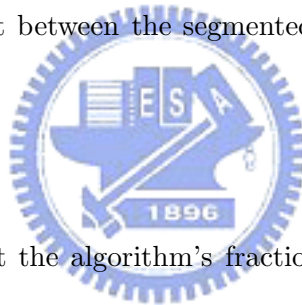


Fig. 3.20: Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask.



Figs. 3.20 and 3.21 present the algorithm’s fractional agreement performance by (2.5) and (2.6) between our segmentation result and the reference mask. The curve in Fig. 3.20 shows that the accuracy of Akiyo sequence achieves 0.996 or better in most frame. Fig. 3.21 demonstrates that most of the the differences of each object’s adjacency tightly followed the standard mask. Tab. 3.1 presents some computing time data for several test sequences: CIF Akiyo, CIF Claire, CIF Salesman, and  $720 \times 480$  Flower Garden (using a personal computer with 1.8-GHz Pentium IV CPU).

Currently, the algorithm treats the two ends of the house appearing behind on the two sides of the tree in the Flower Garden sequence as two objects. It does not recognize that they actually are two parts of one physical object. Staying in the realm of low-level spatio-temporal analysis without incorporating higher-level intelligence, to recognize this fact would

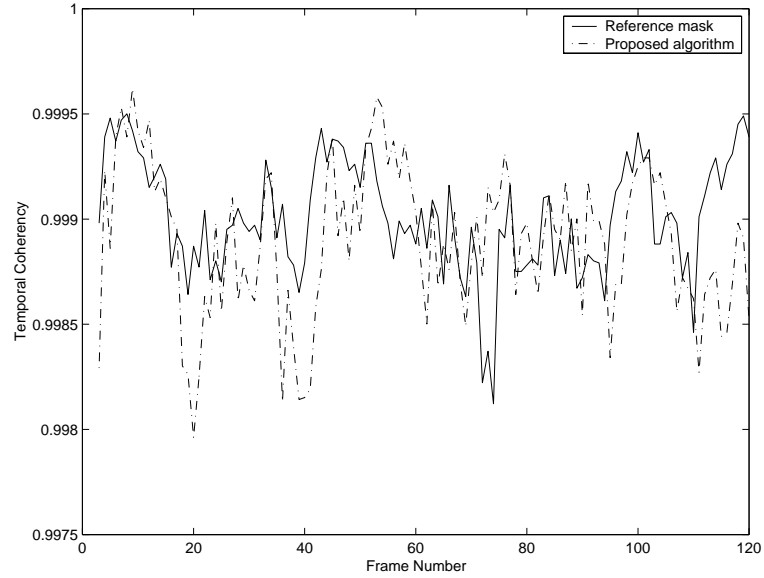


Fig. 3.21: Fractional agreement between the segmented foreground object masks of the Akiyo sequence in adjacent frames.

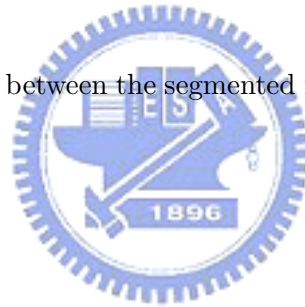


Table 3.1: Average Computing Time in ms per Frame of Major Algorithm Functions

Steps	Pixel Tier	Foreground Tier	Overlays Tier	Total Time
Akiyo	447.3	185.5	42.3	676.6
Claire	455.9	166.4	42.5	664.8
Salesman	493.1	152.7	47.9	693.7
Flower-Garden	1981.2	525.8	194.6	2701.6

require processing on a longer-term memory of the video contents than presently considered. We relegate this to potential future work. Nevertheless, we can perform manual processing of the segmentation result to achieve various effects. For example, one application of video segmentation is video editing based on object manipulation. To demonstrate such use of the segmentation result of our algorithm, Fig. 3.19 shows a synthesized frame using the foreground objects of the Flower Garden sequence with the tree removed.

Compared to some recently published automatic video segmentation algorithms, the present work made a few advances in the following aspects:

1. Some algorithms separate the video into foreground and background only, or into multiple objects with each performing simple motion. Our algorithm can separate multiple overlaid objects doing complicated motion, can deal with object deformation, and can handle object appearance and disappearance.
2. Some algorithms may require advance choice of the processing mode based on the speed of motion in the video, or prior processing of background edges for good performance in fast motion. Our algorithm does not.

## Chapter 4

# Edge-Based Morphological Processing for Efficient and Accurate Video Object Extraction



In the previous two chapters, we used high complexity methods to segment objects. In this chapter, we want to reduce the extraction processes. This chapter considers the edge-linking approach for accurate locating of moving object boundaries in video segmentation.\*

We review the existing methods and propose a scheme designed for efficiency and better accuracy. The scheme first obtains a very rough outline of an object by a suitable means, e.g., change detection. It then forms a relatively compact image region that properly contains the object, through a procedure termed “mask sketch.” Finally, the outermost edges in the region are found and linked via a shortest-path algorithm. The second part of this chapter

---

\*A major part of this chapter has appeared in Yih-Haw Jan and D. W. Lin, “Edge-based morphological processing for efficient and accurate video object extraction,” *IEICE Trans. Information & Systems*, vol. E88-D, no. 2, pp. 335–340, Feb. 2005.

is bi-directional motion estimation for reliable object tracking. We discussed it in Ch. 3. In this chapter we study a multi-tier processing which concerns the drawing out of. The ideas employed in mask sketch reduce the search area effectively. Experiments show that the scheme yields good performance. We develop two edge-linking techniques in this and next chapters.

## 4.1 Background

Extraction of semantic video objects from natural video is a prerequisite for various content-based video applications, and it has attracted much recent attention. Here two key issues are the accurate identification of object boundaries and the required processing time.

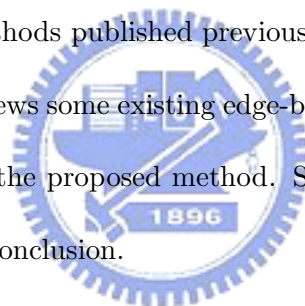
A common design of current object extraction algorithms is to first obtain, roughly, the location and the shape of the objects of interest via spatial and/or temporal analysis and then try to obtain a refined estimate of the object boundaries. In the spatial/temporal analysis, the algorithms may partition the video into regions showing homogeneity in certain features (such as intensity, color, and/or motion), or they may identify image areas showing heterogeneity (such as edges and/or changed areas) that may characterize object boundaries. In the refinement of object boundaries, some common approaches are contour evolution [33, 50], watershed analysis [10, 58], and edge linking [31, 37]. This letter considers the last approach and proposes a way for efficient and accurate extraction of the object boundary, given the object's rough location and shape.

In the edge-linking approach (as well as many other approaches), object boundaries are assumed to be situated at locations showing high intensity or color gradients. A variety of methods can be used to find such locations, for which a popular choice is the Canny edge detector [5]. The need for edge linking arises because typical edge detectors often yield

segments of unconnected contours at object boundaries. Two problems that must be solved in edge linking are thus: 1) among all the edges that can be found in an image, which ones should be considered candidates for linking, and 2) among all the ways the candidate edges can be linked, what is the most proper way of linking (perhaps with slight modification of the edge locations if appropriate).

The two problems above are interrelated, for if we do not suitably limit the range of the candidate edges (the first problem), then the linking (the second problem) will have to trade between computational complexity and accuracy in object delineation. The primary novelty of this work consists in proposing an efficient technique to limit the range of candidate edges so as to facilitate the use of a linking algorithm that can trace the object boundaries closely at a lower complexity than methods published previously.

In what follows, Sect. 2 reviews some existing edge-based segmentation methods and their problems. Section 3 describes the proposed method. Section 4 presents some experimental results. And Sect. 5 is a brief conclusion.



## 4.2 Existing Methods

Given a set of edges in a region, one common way to obtain a rough outline of the object is by orthogonal scans. In one technique [2, 27], each row that contains edge pixels is considered. The space between the leftmost and the rightmost such pixels is filled in. Likewise, for each column that contains edge pixels, the space between the topmost and the bottommost such pixels is filled in. Then a rough object mask is obtained by ANDing the two pixel maps. In another technique [37, 40], a row scan is performed as above. Then the result is subjected to a column scan, whose result is subjected to a second row scan. For convenience, we refer to these techniques as minimal scan and maximal scan, respectively.

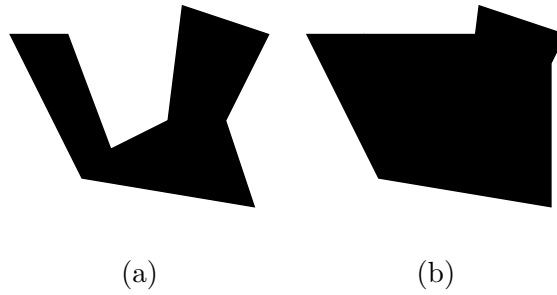


Fig. 4.1: Example of orthogonal scans, assuming perfect edge detection that finds all the boundary pixels of the object. (a) Object shape. (b) Resulting mask of maximal scan.

Ideally, one would desire that the obtained mask boundary be close to the actual object boundary. However, whether this can be the case depends on the object geometry. Consider the object shape illustrated in Fig. 4.1(a), for example, and assume perfect edge detection which finds all the boundary pixels of the object. Then it is not difficult to see that the object mask obtained from minimal scan coincides with the real object shape perfectly while that from maximal scan is as shown in Fig. 4.1(b). For another example, consider the object shape illustrated in Fig. 4.2(a). Minimal and maximal scans would result in the masks shown in Figs. 4.2(b) and (c), respectively. Both expand significantly beyond the true object boundary. A moment's thought reveals that, for maximal scan, the resulting object mask will expand if the object shape is nonconvex and, for minimal scan, it will happen when there exists a cave-shaped area where the cave opening narrows towards its mouth.

After the orthogonal scans, several ways may be used to refine the object mask. Some employ morphological operations [2, 27] (but the details are not clearly given in these publications). And some employ a shortest-path algorithm to find and connect the boundary edges [37, 40]. The first way may yield an enlarged object contour beyond the actual object boundary (in addition to that due to orthogonal scans). The second way follows the edges

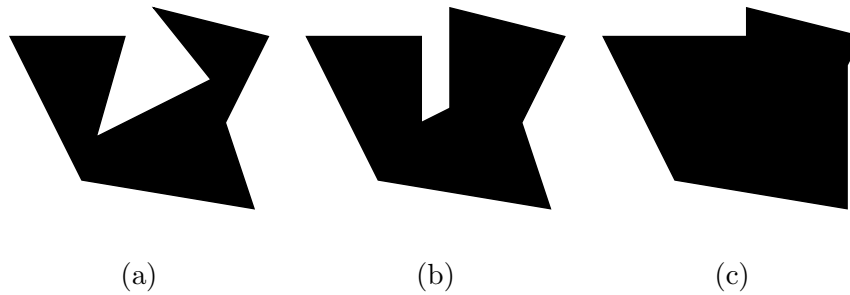


Fig. 4.2: Another example of orthogonal scans, assuming perfect edge detection that finds all the boundary pixels of the object. (a) Object shape. (b) Resulting mask of minimal scan. (c) Resulting mask of maximal scan.

better, and a favorite shortest-path algorithm is Dijkstra's algorithm [13]. However, if the orthogonal scans result in a greatly expanded object mask, then the shortest-path algorithm will need to sift out many pixels, which presents a complexity concern. (For example, a typical implementation of the Dijkstra algorithm has  $O(n^2)$  complexity, where  $n$  is the number of pixels in the search area [49, 53].) Worse yet, if strong edges exist in the overgrown area of the mask and they are not identified and excluded for edge linking through some means, then these edges may be mistaken to be part of the object boundary.

We remark that, while the minimal scan may give a closer outline of the object than the maximal scan when the edges are well-detected, it may yield a grossly distorted object outline when some edges are missed. For example, consider the case where the the extreme lower-right edges of the objects in Figs. 4.1(a) and 4.2(a) are missed by the edge detector. Then the results of maximal scan will stay the same, but the minimal scan will yield the results shown in Fig. 4.3. Therefore, whether one method is better than the other may be case-dependent. Our aim is an efficient scheme that combines the merits of the minimal and the maximal scans.



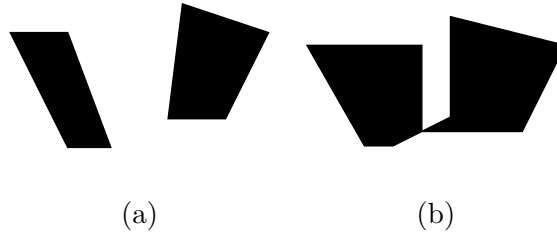


Fig. 4.3: Results of minimal scan when the lower-right object edge is missing. (a) Object of Fig. 4.1(a). (b) Object of Fig. 4.2(a).

### 4.3 The Proposed Method

As indicated previously, we assume that the rough location (in Sec. 3.2.2) and the rough shape of the object of interest have been obtained by some means. Our goal is to identify the object boundary accurately and efficiently through edge linking. The proposal is mainly built on two relatively simple ideas:

1. to make more effective use of the object's known approximate location and shape to narrow the search area, and
2. to make more effective use of the detected edges interior to the search area.

To facilitate algorithm development and the following discussion, we assume use of change detection to roughly delineate the moving objects, but other techniques can also be employed. Change detection detects image areas that exhibit significant changes from one video frame to another. Normally, the result (termed “change detection mask” or CDM in short) will consist of pixels from both the moving objects and the background. Many change detection methods have been proposed in the literature, for example, [1, 21, 27, 38]. Usually, changed pixels are determined by thresholding the frame difference or a filtered version of it, where the threshold may be set considering the amount of camera noise in the video. Since change

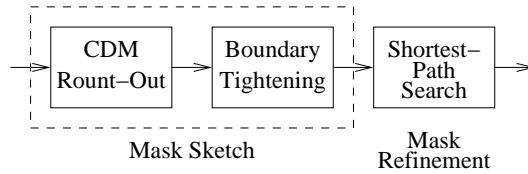


Fig. 4.4: The proposed method of edge linking.

detection only serves an illustration purpose in this work, we omit further discussion of it. Concerning edge detection, we employ the Canny edge detector.

Our edge linking method consists of two stages: mask sketch and mask refinement. In the former we define the outer perimeter of the area that contains an object of interest, and in the latter we refine the estimated object boundary. They are discussed in separate subsections below. The overall procedure is illustrated in Fig. 4.4.

### 4.3.1 Mask Sketch

We illustrate the procedure using the arbitrary CDM example shown in Fig. 4.5(a), where pixels in the CDM are marked in gray. In some cases, a CDM may enclose a half-open area. A common example is a person’s upper body in a videophone scene. In such a case, we consider all pixels between the two farthest separated edge pixels on the frame boundary on the open side of the CDM as edge pixels and include them in the CDM, similar to [27] and [2].

To start, we round out the CDM to obtain a solid region. In addition to defining the maximum support of the object, this step also serves two functions. First, by this we make the edges interior to the CDM, but not part of it, also available for subsequent edge-based processing. And secondly, we stop one- and two-pixel wide “cracks” in the CDM. More than one way exists to obtain the same result. One of them is as follows.

A row scan is performed over each row of the CDM to fill in the space between the two farthest separated pixels in this row. A similar column scan is performed over each column of the CDM. The two results are ORed together to yield an “OR map.” Then we erode the OR map from the outer side inwards so that pixels that are not an eight-connected neighbor of any pixels of the CDM are dropped. By the above we not only fill in the hollow areas inside the CDM (if any), but also stops all one- and two-pixel wide cracks as desired. However, it also expands the CDM slightly by one pixel around. Therefore, we now go one round over the pixels on the outer boundary of each connected set of the already eroded OR map, and delete the pixels that are not in the CDM as we go. This makes the pixel map’s footprint fit the CDM’s shape while keeping the one- and two-pixel cracks filled. The largest connected region of the pixel map is taken as the rounded CDM. For the arbitrary CDM example of Fig. 4.5(a), we obtain Fig. 4.5(b) as the result. Further, it can be seen that the rounded CDM will contain the edges interior to but not part of the original CDM, making them also available for subsequent edge-based processing. Next, we tighten the boundary of the rounded CDM by working with the edge pixels therein. Note that each row of pixels in the rounded CDM may consist of more than one connected segment, and likewise each column. We do “segmental orthogonal scans” as follows. First, for each connected horizontal segment that contains two or more edge pixels, we connect the furthest two of them. Then we regard the boundary pixels in the result as virtual edge pixels and, for each connected segment in each column of the rounded CDM, we connect the two furthest edge pixels. This completes the boundary tightening step. For the above example, let the edge pixels in the rounded CDM be as shown in black in Fig. 4.5(c). Then the resulting pixel maps after segmental horizontal and vertical scans are as illustrated in Figs. 4.5(d) and (e), respectively. To further appreciate the effects of these scans, Fig. 4.5(f) shows the result again, with the edge pixels marked

in black while the others in gray. Comparing it with Fig. 4.5(c), we see that the mask is indeed tightened to match the edge contours better. Note also that, while the technique of orthogonal scans may look much like that in maximal scan, the segmental nature leads to a very different result.

### 4.3.2 Mask Refinement

Now that we have bounded the outer perimeter of the object of interest, we proceed to refine the estimated object boundary. For this we employ Dijkstra's shortest-path algorithm to find and to link up the outermost edges in the boundary-tightened mask. For simplicity, we consider all edges equally as in [37]. Experience indicates that this yields reasonable results. If desirable, the edge strengths (such as magnitudes of gradient values at edge pixels) can be easily incorporated into the path metric, e.g., as in [40], but the effect of noise on the computed values of the edge strengths must be taken into account.

Figure 4.6 illustrates how our algorithm works using a section of the resulting mask shown in Fig. 5.5(f) from mask sketch. Consider edge linking between points  $A$  and  $B$  shown in Fig. 4.6(a), for example. The algorithm considers all edge pixels on the boundary of the mask as belonging to the object boundary, where a pixel is considered to be on the mask boundary if at least one of its eight-connected neighbors is not in the mask. First, all nonedge boundary pixels in the mask are identified. In Fig. 4.6(a), the nonedge boundary pixels between points  $A$  and  $B$  are marked by cross hatching. Next, we stop all edge gaps around the mask boundary that are only one pixel wide. This is done by examining each nonedge boundary pixel. If two of its orthogonal four-connected neighbors are edge pixels, it is declared to be an edge pixel. Fig. 4.6(b) shows the result for the example, where all pixels on the mask boundary (between  $A$  and  $B$ ) that are now considered belonging to the object

boundary are marked black. The others remain cross-hatched. For clarity, in this figure we omit the black marking of the edge pixels that are not on the mask boundary. Note that the above gap-stopping method is in effect a kind of shortest-path algorithm for edge linking over one-pixel gaps, but with reduced complexity compared to the Dijkstra algorithm in normal operation. Regarding the example, we are now left with two edge discontinuities between  $A$  and  $B$ , defined by the pixel pairs  $(a, b)$  and  $(c, B)$ , respectively.

The algorithm continues by considering separately each remaining edge discontinuity along the mask boundary. For each discontinuity, we search in the mask for the shortest path that bridges it, where each edge pixel in the mask is given an equivalent length  $d_0$  and each nonedge pixel an equivalent length  $d_1$ . (We let  $d_0 = 1$  and  $d_1 = 10$  in the experiments.) To control the computational complexity, we may limit the search area to a band around the mask's boundary. This is equivalent to assuming that the object boundary be within the width of the band. Let  $D_w$  be the bandwidth in number of pixels. For example, Fig. 4.6(c) illustrates the two search areas for the edge discontinuities  $(a, b)$  and  $(c, B)$ , respectively, with  $D_w = 5$ . After executing Dijkstra's shortest-path algorithm over the two search areas separately, we obtain the final result shown in Fig. 4.6(d) for edge linking between  $A$  and  $B$ .

### 4.3.3 Remarks on Complexity

The complexity of the above algorithm is difficult to characterize precisely, because it depends on the detailed organization of the operations involved. Nevertheless, we can see that an unsophisticated implementation of mask sketch that follows the description given above may involve several passes over the CDM and its interior, each pass involving some simple logical operations on each pixel. Therefore, the complexity of mask sketch is on the order of the size of the extracted object. The complexity of mask refinement depends on the total length

of the edge discontinuities. We mentioned that a typical implementation of the Dijkstra algorithm has  $O(n^2)$  complexity, where  $n$  is the number of pixels in the search area. Thus the complexity of mask refinement is at most  $O(L^2 D_w^2)$  where  $L$  is the perimeter of the extracted object.

In contrast, the maximal and the minimal scans both require complexities of a similar order-of-magnitude to mask sketch. However, since their results do not follow the object shape as closely as the mask sketch's, if a shortest-path algorithm is used subsequently to delineate the object better, the algorithm may need to search over a considerably bigger area for the object boundary, which means potentially significant complexity penalty. Some examples are given in the next section.

## 4.4 Experimental Results

To illustrate the working and the performance of the proposed algorithm, we present some results for a typical frame in the CIF test sequence Mother and Daughter. (Results for other frames are similar in nature.) The sequence gives a typical example of the situation where the moving objects have a grossly nonconvex contour. Figure 4.7 shows an original frame, the corresponding CDM, the rounded CDM, and the result of mask sketch. Figure 4.8 shows the results of mask refinement at  $D_w = 2$  and  $D_w = 5$ . The two results differ only slightly and both are subjectively pleasing.

For comparison, Fig. 4.9 shows the result from using maximal scan on the CDM. Due to the greatly expanded object mask in cave-shaped areas from maximal scan, compared to the proposed method the subsequent shortest-path algorithm needs to search over a much larger area for the correct object boundary. In contrast, minimal scan usually results in less expansion of the object mask than maximal scan and is thus of a lesser complexity concern

when its result is used as the starting point for edge linking by shortest-path search. But nevertheless the tighter mask area resulting from the proposed method still offers a complexity advantage in edge linking. Figure 4.10 shows the results of mask refinement at  $D_w = 2$  and  $D_w = 5$  under minimal scan. Note the difference in the face region of the mother compared to the corresponding results in Fig. 4.8. At  $D_w = 7$ , the result becomes better aligned with the true object contour in this face region. (We remark that, as noted previously, [27] and [2] do not employ a shortest-path algorithm to obtain the final object mask. Both employ morphological operations, but the details are not clearly given. Both result in an enlarged object mask in cave-shaped areas of the object. Our focus in this work is not to compare the performance of edge linking with these methods, but to obtain an efficient edge linking scheme with good performance.)

To assess more quantitatively the relative complexity of different methods, we define a measure called *refinement efficiency (RE)* as

$$RE = \frac{NC}{ND}, \quad (4.1)$$

where  $NC$  is the number of object boundary pixels filled in by Dijkstra's algorithm and  $ND$  is the total number of pixels examined by the algorithm in the process. Note that  $RE$  is a function of  $D_w$ . Generally speaking,  $RE$  should tend to decrease as  $D_w$  increases. Moreover, assuming that the object mask obtained at  $D_w = \infty$  is the true mask, we define a measure called *mask convergence (MC)* as

$$MC = 1 - \frac{\sum O_W \oplus O_\infty}{O_\infty}, \quad (4.2)$$

where  $\oplus$  denotes exclusive OR (XOR) and  $O_W$  and  $O_\infty$  are the object masks obtained with  $D_w = W$  and  $D_w = \infty$ , respectively. It measures the closeness of  $O_W$  to  $O_\infty$ , and its value is always smaller than or equal to 1. It is equal to 1 only when  $O_W$  and  $O_\infty$  are identical. Jointly

considering  $RE$  and  $MC$  gives us a better picture of the algorithm performance. Having high  $RE$  and  $MC$  values at low  $D_w$  values indicates efficient and accurate segmentation. In this regard, Fig. 4.11 shows that the proposed method is more efficient in edge linking than processing based on maximal scan or minimal scan.

Furthermore, the objects moves continuously through temporal evolution. When an object moves slowly at a certain time instance, we forward superpose the object's projection and the current change area. For robustness, backward check the superimposed objects and repeatedly use the above method to extract the objects in the latter frames. Details are in Sec. 3.2.1 and [23]. Figs. 4.12 and 4.13 present the algorithm's fractional agreement performance by the (2.5) and (2.6) between our segmentation result and reference mask. The curve in Fig. 4.12 shows that the accuracy of Akiyo sequence achieves 0.995 or better in most frame. Fig. 4.13 demonstrates that most of the the differences of each object's adjacency tightly followed the standard mask. Tab. 4.1 presents some computing time data for several CIF sequences: Akiyo, Claire, Salesman, and Mother-Daughter (using a personal computer with 1.8-GHz Pentium IV CPU).

We considered the edge-linking approach to accurate extraction of object boundaries for natural video segmentation. The primary novelty of the proposed method consists in an efficient technique to limit the range of searched candidate edges for linking and facilitate the use of a shortest-path algorithm that can trace the object boundaries closely at a lower complexity than methods published previously. Experimental results demonstrated that the proposed method was efficient and performed well. A topic for potential future work is further enhancement of the algorithm efficiency. Another is a more comprehensive comparison of the complexity and performance of the edge-linking approach with that of watershed analysis and contour evolution.



Table 4.1: Average Computing Time in ms per Frame of Major Algorithm Functions

Algorithm	Motion	Orthogonal	Mask	Segmental	Edge
Functions	Estimation	Scan	Trimming	Orthogonal Scan	Linking
Akiyo	185	3.9	4.2	3.9	6.49
Claire	156	3.1	6.6	3.7	4.61
Salesman	126	3.4	7.1	3.2	7.07
Mother-Daughter	247	4.4	11.3	4.7	10.8

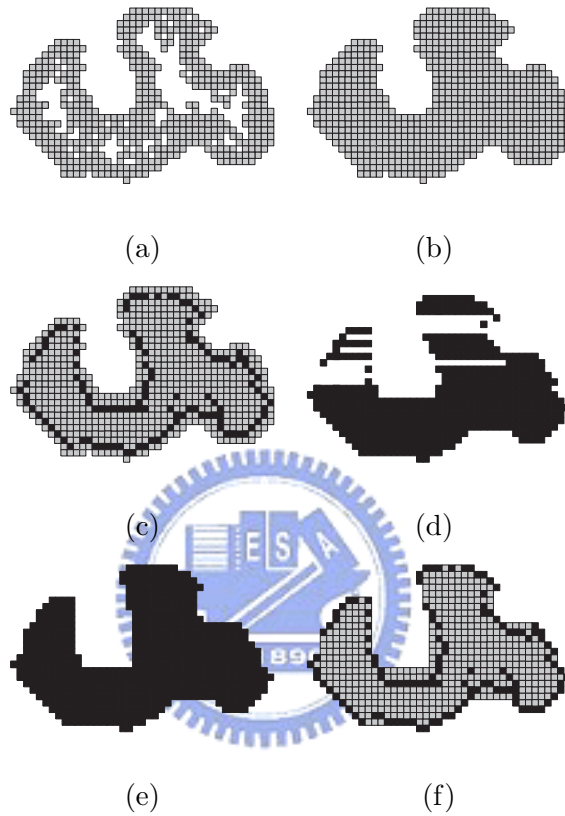


Fig. 4.5: Arbitrary example for illustration of the proposed algorithm. (a) CDM. (b) Result of CDM round-out. (c) Result of CDM round-out, with edge pixels therein marked in black. (d) Result of segmental row scans. (e) Result of segmental column scans, i.e., result of boundary tightening. (f) Result of boundary tightening with edge pixels in the mask marked in black while other pixels marked in gray.

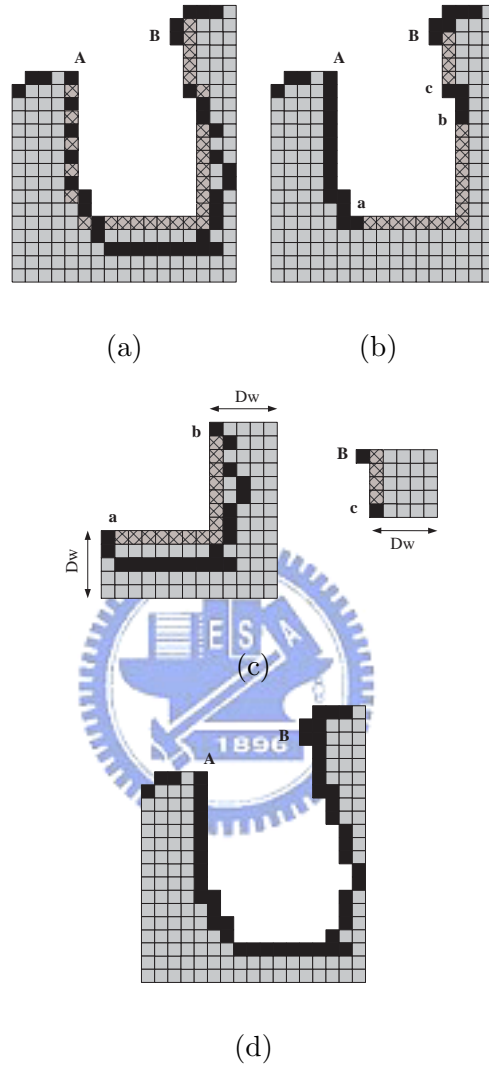


Fig. 4.6: Illustration of the mask refinement method. (a) Zoomed-in section of the mask sketch result for illustration use. (b) After stopping of one-pixel gaps. (c) Respective search areas of shortest-path algorithm for edge discontinuities  $(a, b)$  and  $(c, B)$ . (d) Final result of edge linking between  $A$  and  $B$ .

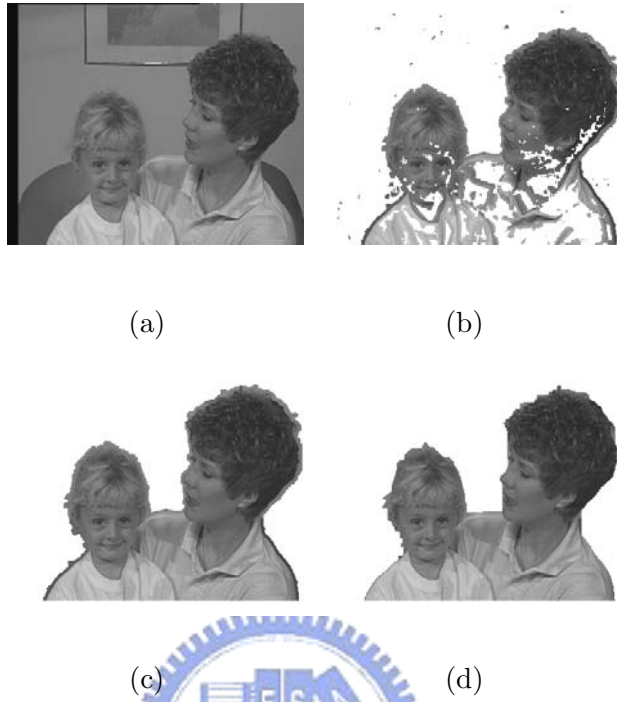


Fig. 4.7: Illustration of algorithm behavior. (a) An original frame of Mother-and-Daughter sequence. (b) CDM. (c) Rounded CDM. (d) Result of mask sketch.

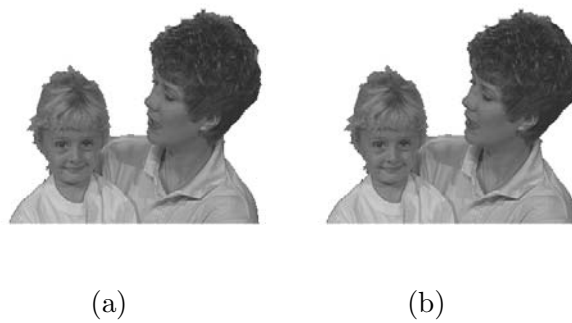


Fig. 4.8: Result of mask refinement at different search bandwidths. (a)  $D_w = 2$ . (b)  $D_w = 5$ .



(a)

(b)



(c)

(d)

Fig. 4.9: Segmentation results for Mother and Daughter based on maximal scan. (a) Result of maximal scan on CDM. (b),(c),(d) Results of mask refinement with  $D_w = 20, 60,$  and  $100,$  respectively.

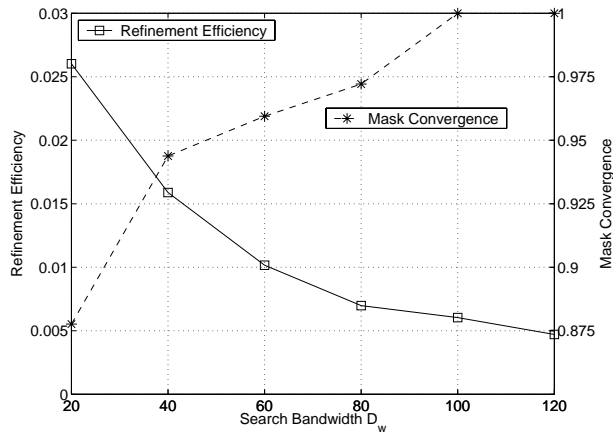


(a)

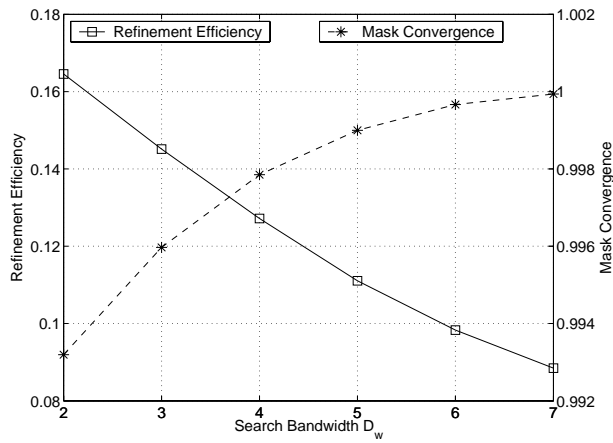
(b)

Fig. 4.10: Result of mask refinement based on minimal scan at different search bandwidths.

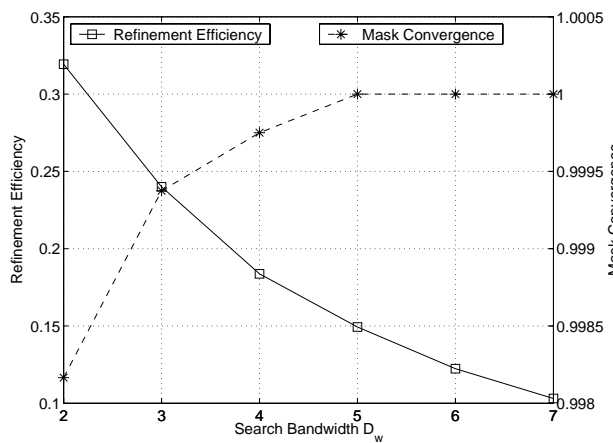
(a)  $D_w = 2.$  (b)  $D_w = 5.$



(a)



(b)



(c)

Fig. 4.11: Comparison of algorithm efficiency and accuracy in segmentation of Mother-and-Daughter sequence. (a) Maximal-scan based processing. (b) Minimal-scan based processing. (c) Proposed method.

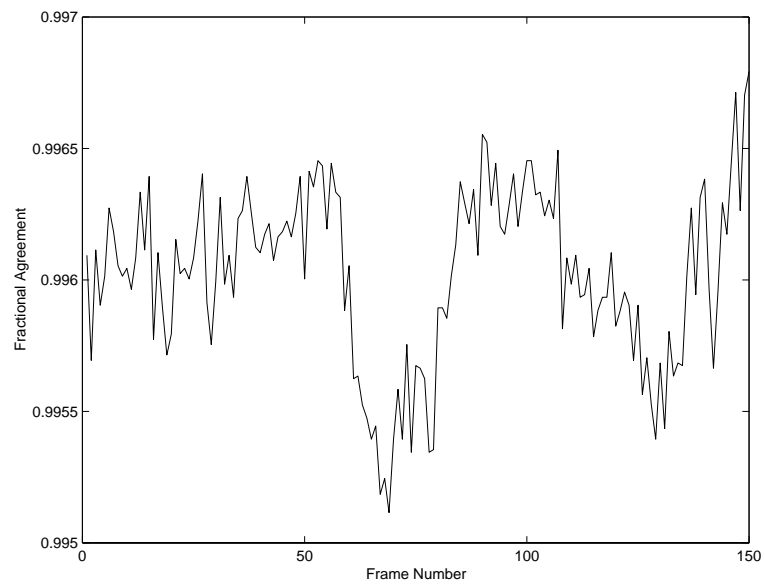


Fig. 4.12: Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask.

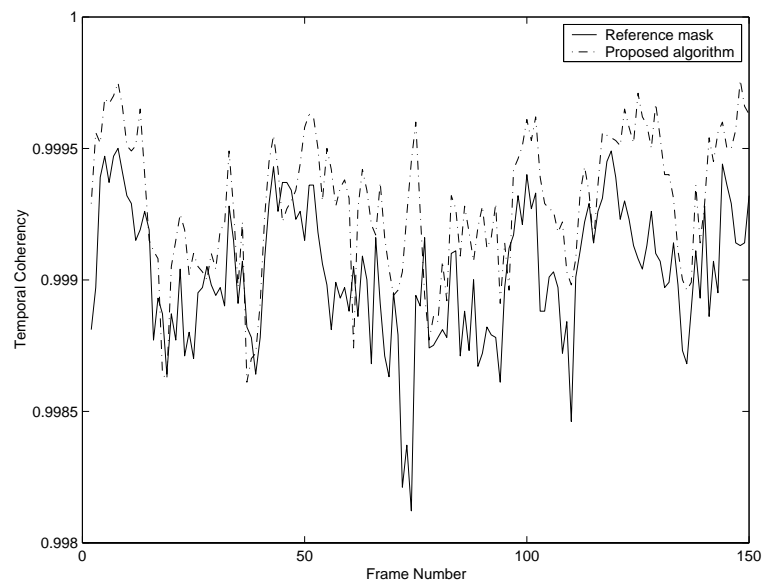


Fig. 4.13: Fractional agreement between the segmented foreground object masks of the Akiyo sequence in adjacent frames.



## Chapter 5

# Reduced-Complexity Motion

# Analysis and Edge Processing for

# Accurate Identification of Object

# Boundaries



In previous chapters, We developed relatively sophisticated segmentation algorithms that can identify moving object boundaries reasonably accurately and can separate multiple overlaid moving objects [22]. In addition, for slow moving objects, we develop an edge-based morphological processing considering efficient and accurate object extraction. This chapter considers accurate delineation of moving object boundaries with reduced complexity.\* By experience, we find that each of the following three image analysis techniques has characteristics useful to

---

\*Part of this chapter will appear in Yih-Haw Jan and D. W. Lin, “Automatic video segmentation with novel motion analysis and edge processing for accurate identification of object boundaries,” in *Int. J. Electrical Engineering*.

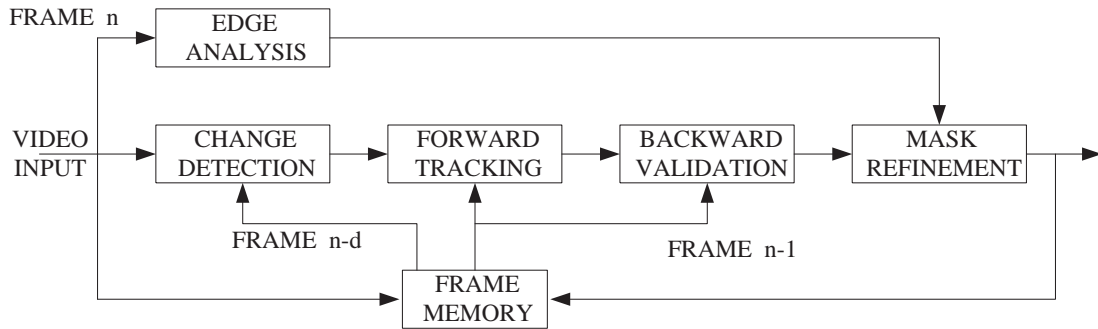


Fig. 5.1: Structure of the proposed algorithm.

the intended task: *change detection* is good at identifying significantly changing areas across video frames, *motion estimation* is good for tracking object motion, and edge detection can provide accurate information concerning object boundaries. Their characteristics are complementary. We choose to found our algorithm on all three techniques jointly. Specifically, we use them to obtain a sketch of the “mask” of the moving objects. And we design a particular method to refine the mask for accurate delineation of the moving object boundaries. The primary novelties of the proposed scheme consist in the mask refinement method and the method of motion estimation with reduced complexity. Fig. 5.1 shows the overall structure of the proposed algorithm. Roughly speaking, semantic objects are detected initially with “change detection,” tracked in time with motion-compensated “forward tracking” and “backward validation” that allow object shape changes, and their boundaries delineated more accurately with “mask sketch.” The block “edge analysis” provides useful information for the last function.

In what follows, Sections 5.1 and 5.2 describe the proposed segmentation algorithm with its novelties, except the edge analysis block which employs the well-reputed Canny edge detector [5]. Then Section 5.3 presents some experimental results. And Section 5.4 gives the

discussion.

## 5.1 Motion-Related Analysis

The design of the change detection (details have introduced in Sec. 3.2.2) and the motion-compensated object tracking functions is based on the following observation. In video signal analysis and segmentation, motion information is helpful for tracking of moving objects. However, conventional motion estimation methods are usually very computation-intensive and need not yield reliable motion information. As a result, one often would like to minimize its use or its complexity. By simply detecting the changed areas, change detection is a favored low-complexity mechanism to roughly identify image regions that contain moving objects [1, 26, 37, 38]. However, if an originally moving object comes to a standstill, change detection will lose track of the object unless a long-term memory is provided, such as in [9]. Moreover, there is also a concern in accuracy of object boundary identification without motion information. In this work, we choose to develop a low-complexity, object-based motion estimation technique for object tracking. It is complemented by change detection which helps in capturing fully the possible temporal changes in object shape and in identifying new moving objects (especially in the first two video frames).

### 5.1.1 Forward Tracking

A main purpose of the “forward tracking” function is to find the footprint of each object of the previous frame (say  $I_{n-1}$ ) in the current frame (say  $I_n$ ). Some inaccuracy is tolerable, because the subsequent backward validation and mask refinement will localize the object boundary more precisely. By integrating the footprint with the change detection output, we can accommodate appearance of new moving objects and handle object deformation better.

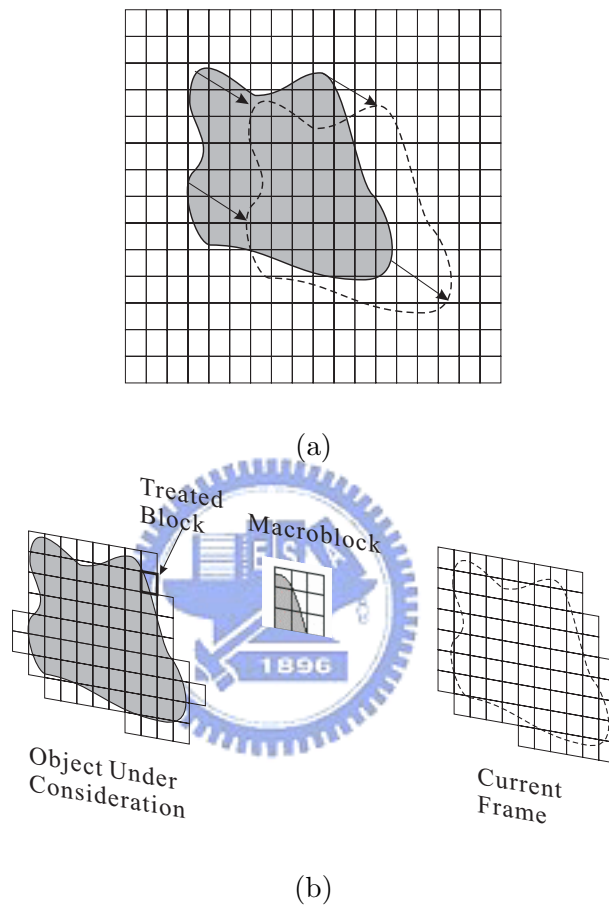


Fig. 5.2: Object-based motion estimation. (a) An object under consideration (shaded region) in the previous frame has moved to a different position in the current frame (dashed contour). Each small square is  $BW \times BW$  in size. (b) Illustrating the idea that motion estimation is carried out on the  $3 \times 3$  macroblock centered at the treated block.

Key in the forward tracking function is a novel low-complexity motion estimation method, which is illustrated in Fig. 5.2. Fig. 5.2(a) shows that an object under consideration in frame has moved to a different position in frame . The object is divided into square blocks of size  $BW \times BW$  pixels. (We have used  $BW = 4$  in this work.) To save computation, we first consider all blocks on the object boundary. For each such block, we perform block-matching motion estimation for a “macroblock” of  $3 \times 3$  blocks centered at this block, as illustrated in Fig. 5.2(b). For simplicity, we only consider translational motion. However, the motion vectors may point out of the frame as in the unrestricted motion vector (UMV) mode of the ITU-T H.263 standard [48]. The required out-of-frame pixels are obtained by repeating the pixel values at frame boundaries. The obtained motion vector is assigned to the pixels of the treated block. We then proceed inwards from the boundary blocks until all the interior blocks are treated. In each step, we treat one “layer” of blocks that are immediately inside the blocks that had been treated in the previous step. For each of these blocks, we perform macroblock-based motion estimation similar to the case of boundary blocks, but the candidate motion vectors are highly limited: only the zero vector and the motion vectors of its already treated neighbors are tested, and the best is taken. An exact evaluation of the complexity of the above motion estimation method is impossible, as it depends on the shape and the size of the object. To gain some idea, consider the situation where the moving foreground objects occupy a rectangular region of 50% of the frame size, for example. Then, for CIF video ( $352 \times 288$ ), there may be about 200 blocks along the perimeter of the region and about 3000 blocks inside it. Since each interior block has 9 neighbor blocks, assume that each of them needs to check 5 motion vectors on the average. Consider maximum motion up to  $\pm 14$  pixels in each direction. Therefore, the proposed method is about 8% and 25% the complexity of purely full-search and three-step search algorithms [48], respectively. Further simplification

of the motion estimation algorithm can be considered, but is left to potential future work. To continue, the mask (i.e., pixel map) of each object in  $I_{n-1}$  is projected forwardly onto using the motion vectors obtained above. For convenience, let  $O_{i,n-1}$ ,  $i = 1, 2, \dots, S$ , denote the  $i$ th object in  $I_{n-1}$  and let  $P_{i,n}$  denote the corresponding projected footprints. The forward-tracked mask  $PCD_{i,n}^F$  of each object is obtained by taking the union of  $P_{i,n}$  and  $CD_n$ , retaining the largest connected set of pixels, and filling up all small isolated “holes” (in the interior of the connected set) that may show up in the pixel map due to slight difference in the estimated motion of nearby blocks.

### 5.1.2 Backward Validation

Backward validation is conducted to trim each mask  $PCD_{i,n}^F$  for better accuracy, since the forward motion estimation need not be very accurate and since  $CD_n$  may contain contributions from more than one object. For this, backward motion estimation from  $I_n$  to  $I_{n-1}$  is performed for the pixels in  $PCD_{i,n}^F$  that are not in  $P_{i,n}$ . The method is similar to the forward motion estimation described above. Only those pixels whose backward motion-compensated projections lie inside or touch  $O_{i,n-1}$  are retained. As in forward tracking, if any small isolated holes show up in the trimmed  $PCD_{i,n}^F$ , they are filled up. The resulting set of validated pixels is denoted  $PCD_{i,n}^B$  and referred to as the coarse mask. In typical videophone scenes, the pixels needing backward validation are relatively few. Hence the required backward motion estimation does not add much computation.

## 5.2 Mask Refinement

Due to the nature of our change detection and motion estimation methods, the coarse mask  $PCD_{i,n}^B$  may contain background pixels beyond the actual object boundary. The mask re-

finement function seeks to obtain a better localization of the object. Since object boundaries are often characterized by high gradients in pixel values, our mask refinement procedure does edge-based processing. For natural video, typical edge detectors (such as the Canny detector that we use [5]) often do not obtain closed or connected contours at object boundaries. Therefore, in addition to finding the correct edges that mark object boundaries, a way to connect the “broken” boundary edges must be devised.

We assume that an object is located completely within the coarse mask. In many cases, a coarse mask may enclose a half-open area. A common example is a person’s image in a videophone scene, which may show the person’s upper body butted to the bottom side of the frame and result in a coarse mask that is half open on the lower side. In such a case, we consider all pixels between the two farthest separated edge pixels on the frame boundary on the open side of the mask as edge pixels and include them in the mask, similar to [27].



Fig. 5.3 shows a flowchart of our mask refinement procedure. It includes four function blocks. The “mask rounding” block rounds out the coarse mask to form a solid pixel map. The result enables us to tell the outer side from the inner side of the pixel map. It also enables us to define the outermost edge pixels in subsequent processing. The “footprint tightening” block trims away some possibly existing overgrowth on the outer side of the rounded mask. The “boundary sharpening” block further shrinks the mask so as to locate the object boundary at existing edges where suitable and to fill in the gaps that may be present between these edges. And lastly, the “mask tuning” block puts together the output of the boundary sharpening block and fine-tunes it to yield the final refined mask.

We will illustrate the procedure using the arbitrary coarse mask example shown in Figure 4. This mask is highly non-convex on the outer side to signify that a distinguishing feature of the proposed method is the ability to handle highly non-convex object shapes well. In

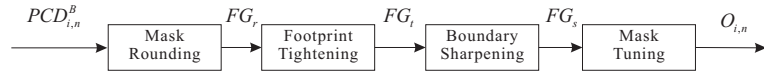


Fig. 5.3: Mask refinement procedure.

addition, the mask is grossly hollow, which is a non-ideal situation that is not expected to happen often (except for the first frame of a video sequence where the mask refinement must be based on the change detection output) but can be dealt with by the proposed procedure.

### 5.2.1 Mask Rounding

To start, the “mask rounding” function does orthogonal scans. Specifically, a horizontal scan is performed over each row in the coarse mask to fill in the space between the leftmost and the rightmost pixels. Corresponding vertical scans are performed to obtain another result. The union of the two results is taken. The above closes up the interior of the mask, but the obtained union map may be larger than the actual object. (Fig. 5.5(a) shows the result corresponding to Fig. 5.4(a).) To cut away the overgrowth, we erode the union map from the outer side inwards, so that every outer pixel that is neither in the coarse mask nor an eight-connected neighbor of it is removed. (Fig. 5.5(b) shows the corresponding result for the illustrative example.) This will also stop all one- and two-pixel “cracks” on the outer side of the coarse mask, but will in general cause the coarse mask to grow by one pixel around. We thus further examine the outermost “slice” of pixels and remove all that are not in the original coarse mask. The result, denoted  $MR_{i,n}$ , is a solid area enclosed by the coarse mask, with one- and two-pixel cracks filled up. The complexity of this work is roughly on the order of the size of  $MR_{i,n}$ , for there may be up to several index computations, memory accesses, and comparisons performed on each pixel.



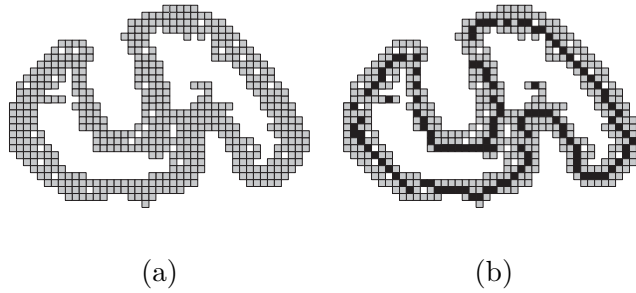


Fig. 5.4: An arbitrary example for illustration of the mask refinement method. (a) The coarse mask (gray pixels). (b) The coarse mask with the edge pixels therein marked in black.

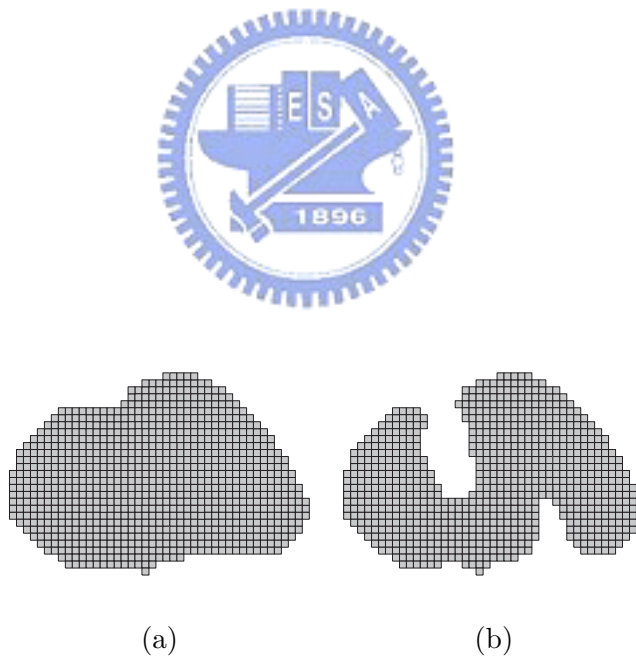


Fig. 5.5: Illustration of the mask rounding procedure. (a) The mask after orthogonal scans and taking the union. (b) The eroded mask ( $FG_r$ ).

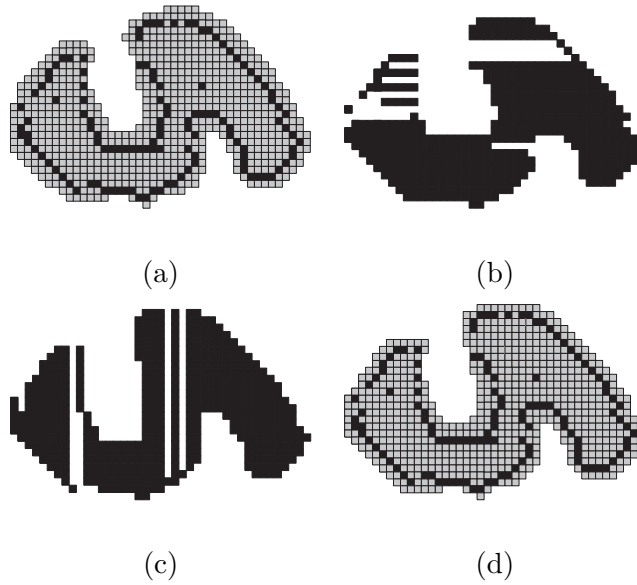


Fig. 5.6: Illustration of the footprint procedure. (a)  $FG_r$  with the edge pixels therein marked in black. (b) Row map. (c) Column map. (d) The footprint-tighten mask ( $FG_t$ ), with edge pixels therein marked in black.

### 5.2.2 Footprint Tightening

We assume that most edge pixels close to the perimeter of  $MR_{i,n}$  define the object boundary. We now tighten the footprint of  $MR_{i,n}$  somewhat to make its boundary closer to the outermost edge pixels in it. For this, we first examine each connected horizontal and vertical line segment in  $MR_{i,n}$ . For each horizontal line segment, if the outermost edge pixels are close to the ends of the segment (say within several pixels of an end and sufficiently distant from the segment's center), then the space between them is filled. Call the result a row map. A similar operation is carried out vertically to result in a column map. For example, let the edge pixels in the  $MR_{i,n}$  of Fig. 5.5(b) be located as shown in black in Fig. 5.6(a). Then the row and the column maps are as shown in Fig. 5.6(b) and (c), respectively.

Next, we trim the  $MR_{i,n}$  from the outer side inwards. Any outer pixel that is not an

eight-connected neighbor of the row map or the column map is removed. We name the result  $FT_{i,n}$ . For convex parts of the object, the outer boundary of  $FT_{i,n}$  is at most two pixels away from an edge pixel horizontally and vertically. Fig. 5.6(d) illustrates the result obtained for the arbitrary example. The complexity of footprint tightening is also roughly on the order of the size of  $MR_{i,n}$ , or equivalently, roughly on the order of the size of  $FT_{i,n}$ .

### 5.2.3 Boundary Sharpening

Based on  $FT_{i,n}$ , we now define the object boundary more definitely. This block contains two steps: the first, termed *overgrowth pruning*, locates the object boundary at existing edges where suitable, and the second, termed *edge filling*, interpolates between the assumed boundary edges where gaps exist.

In overgrowth pruning, we go over the boundary pixels in  $FT_{i,n}$ . For each pixel, if an edge pixel exists within a distance of  $B$  pixels in the  $FT_{i,n}$ , then we prune the pixels from the boundary pixel inwards, stopping at the edge pixel. Otherwise, no action is taken. This is carried out in the horizontal and the vertical directions separately. From previous discussion on footprint tightening, a suitable value for  $B$  may be 3. Continuing on the illustrative example, Figs. 5.7(a) and (b) show the results after this step. The pixels marked “X” are those boundary pixels at which no pruning action is taken. They indicate where edge interpolation needs to be carried out.

The above processing invites a question, namely, it seems that we have kept some pixels outside the edge pixels in footprint tightening, only to be pruned in this step. The answer is that, by keeping these pixels in footprint tightening, we also maintain some additional latitude in placement of the interpolated edges in gaps.

Now in edge filling, we conduct edge interpolation by examining the boundary pixels

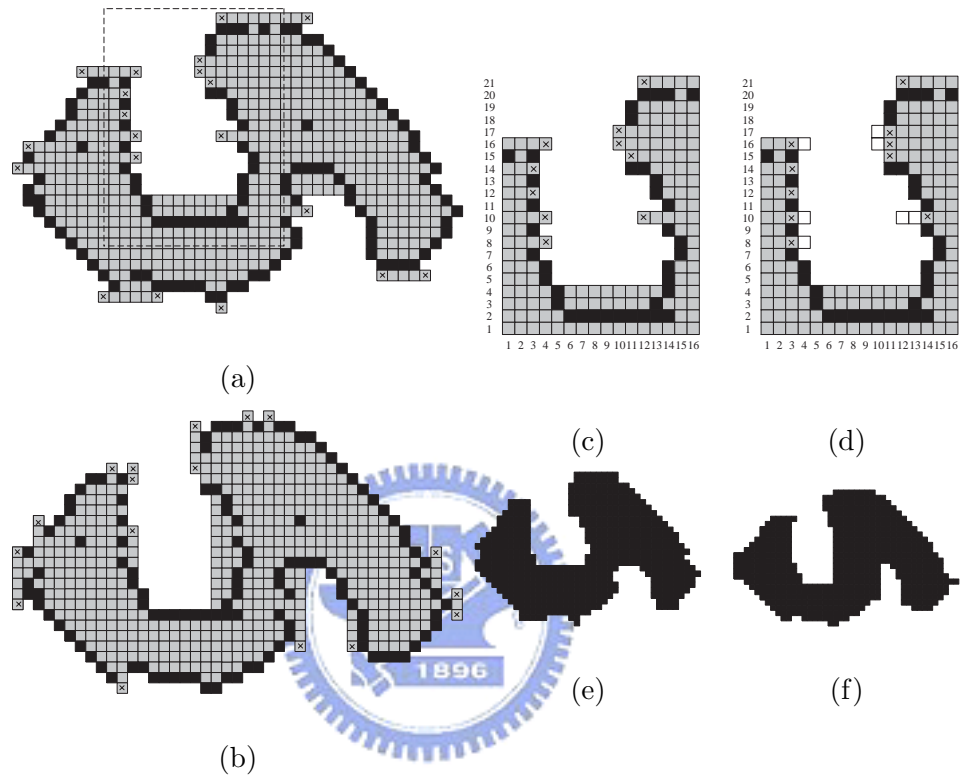


Fig. 5.7: Illustration of the boundary sharpening procedure. (a) Mask after overgrowing pruning in the horizontal direction. (b) Mask after overgrowing pruning in the vertical direction.. (c) A section of the horizontally pruned mask. (d) The same section after edge filling in the horizontal direction. (e) Converged result of overgrowth pruning and edge filling in the horizontal direction. (f) Converged result of overgrowth pruning and edge in the vertical direction.

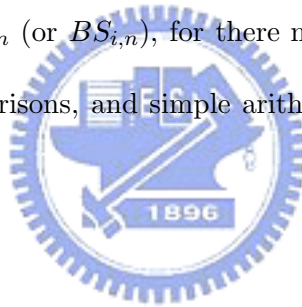
where the pruning action has been skipped in the first step. Again, the processing in the horizontal and the vertical directions are performed separately. We describe the processing in the horizontal direction below. The processing in the vertical direction is similar.

For illustration, Fig. 5.7(c) shows a zoomed-in plot of a section of the object in Fig. 5.7(a) as indicated there in dashed lines. Call the pixels marked “X” *potential edge pixels*. Consider each of them in turn in raster-scan order. Let  $C_x$  denote the pixel that is being examined currently. We check if an edge or a potential edge pixel exists within a horizontal distance of  $B$  pixels in the line segment immediately above. If so, then the distance with the closest one is noted, where the distance is measured in orthogonal steps that one pixel has to take to reach the other pixel. Let  $C_u$  denote this nearest edge or potential edge pixel above. In the same way, the nearest edge or potential edge pixel below, say  $C_d$ , is found. Then the mean column position between  $C_u$  and  $C_d$  is obtained. The potential edge pixel  $C_x$  is moved to that column position, adding a further 0.5 steps if that column position happens to be a fractional one. Several exceptional cases are as follows. First, if a  $C_u$  or a  $C_d$  as defined above cannot be found, then we take its column position to be the same as  $C_x$ . Second, if the new pixel position for  $C_x$  is outside , then  $C_x$  is not moved.

For example, consider the potential edge pixel at  $(H,V) = (10,17)$  in Figure 7(c), where  $H$  denotes the horizontal coordinate and  $V$  the vertical coordinate of the pixel. Let  $B = 3$ . The nearest edge or potential edge pixel above, within  $B$  pixels horizontally, is at  $(11,18)$  with distance 2. The nearest edge or potential edge pixel below, within  $B$  pixels horizontally, is at  $(10,16)$  with distance 1. Thus the potential edge pixel is moved to  $(11,17)$ . This new position for the potential edge pixel is used in subsequent processing. For example, for the potential edge pixel at  $(10,16)$ ,  $C_u$  and  $C_d$  are located at  $(11,17)$  and  $(11,15)$ , respectively. It is moved to  $(11,16)$ , therefore. The overall result is as shown in Fig. 5.7(d).

One can iterate over the two steps of overgrowth pruning and edge filling several times to stabilize the masks fully. But experience shows that one pass usually suffices; more iterations affect the masks only very slightly. Some real video examples will be given later. For the illustrative example, a second iteration in the horizontal direction yields no changes on the mask at all. A second iteration in the vertical direction merely removes, in overgrowth pruning, the remaining gray pixels in the third column on the right-hand side of the mask in Fig. 5.7(b) that have survived the first iteration. The resulting masks are shown in Fig. 5.7(e) and (f). The output of the boundary sharpening function, termed  $BS_{i,n}$ , consists of the two masks resulting from the horizontal and the vertical processing.

The complexity of the boundary sharpening function, apparently, is on the order of the length of the perimeter of  $FT_{i,n}$  (or  $BS_{i,n}$ ), for there may be up to several index computations, memory accesses, comparisons, and simple arithmetic operations conducted for each boundary pixel.



#### 5.2.4 Mask Tuning

We now obtain the final refined mask from the two directional masks yielded by boundary sharpening. In this, we first take the intersection of the two masks in  $BS_{i,n}$ . For the illustrative example, the intersection of Fig. 5.7(e) and (f) is as shown in Fig. 5.8(a). We then fine-tune the boundary slightly to smooth out some rough turnings. This is achieved by examining the layer of pixels just outside the intersection map. For each pixel, if a large number (say 5 or more) of its eight-connected neighbors are in the intersection map, then we add this pixel to the mask. The resulting mask defines the extracted object  $O_{i,n}$ . Fig. 5.8(b) shows the result for the illustrative example. The complexity of mask tuning is on the order of the size of  $BS_{i,n}$  (or  $O_{i,n}$ ).

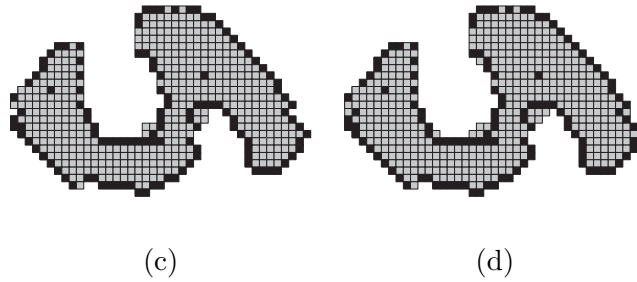


Fig. 5.8: Illustration of the mask tuning procedure. (a) Intersection of the two directional masks in  $FG_s$ . (b)  $O_{i,n}$ .

### 5.2.5 Summary and Comparison with Other Edge-Based Methods

In summary, the above mask refinement procedure makes several passes over the mask area and several passes over its boundary. Each pass may involve up to several index computations, memory accesses, comparisons, and/or simple arithmetic operations for each traversed pixel. The precise amount of computation depends on the detailed organization of the program and is difficult to characterize exactly. We shall provide some information on the computation time used by our (unoptimized) program on several common test sequences in the next section.

For comparison, consider two other groups of edge-based methods for mask refinement (in our terms). One of them takes the union (in some sense) of orthogonal scans [37, 40], and the other the intersection [2, 27].

The first group of methods, by their way of doing orthogonal scans, will fill up the outer indents (i.e., non-convex regions) in a way resembling what our mask rounding does initially (see Fig. 5.5(a)). Then a shortest-path algorithm is used to find and connect the boundary edges in the resulting mask, where the most popular shortest-path algorithm is the Dijkstra algorithm [13]. Because Dijkstra algorithm's complexity is typically  $O(N^2)$  where  $N$  is the

size of the search area [49, 53] the search over large indents for boundary edges can be very computation intensive. In contrast, the complexity of our algorithm, as analyzed above, is  $O(N)$ .

The second group of methods have complexity  $O(N)$  and are computationally simpler than our algorithm. By taking the intersection of the resulting masks of orthogonal scans, they will fill up the outer indents of a mask to a lesser degree than the first group of methods. This is illustrated in Fig. 5.9, where Fig. 5.9(a) and (b) show the results of orthogonal scans that fill in the space between the outermost edge pixels in the coarse mask horizontally and vertically, respectively, and Fig. 5.9(c) shows their intersection. The thin white stripes protruding into the intersection map as a result of broken edges around the mask boundary can be filled up with basic morphological operations such as closing. But the overgrowth outside the boundary edges where the object mask is non-convex cannot be handled satisfactorily with these basic operations. Indeed, some examples in [27] (Fig. 5.3(d)) and [2] (Fig. 5.3(d)) show that the segmented object masks leave the indents filled, yielding an inaccurate segmentation where the object contour is non-convex. It is very common for real-life objects to have non-convex contours. The proposed algorithm can handle this situation more satisfactorily.

### 5.3 Experimental Results

We present some experimental results on several common CIF test sequences to illustrate the working and the performance of the proposed video segmentation algorithm. We first give a walk-through of some key algorithm steps. Then we present some segmentation results. And lastly, we provide some data regarding the algorithm speed.



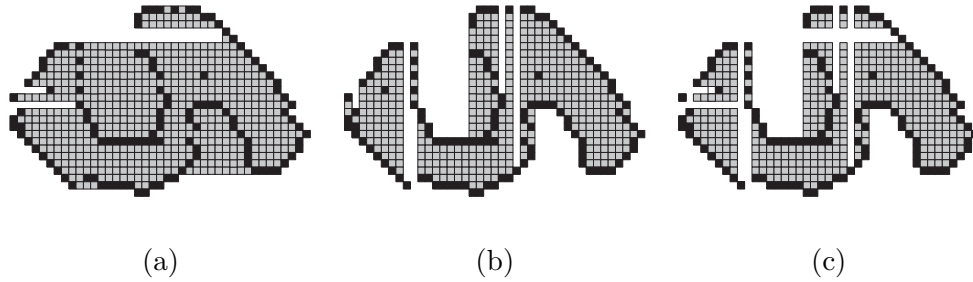


Fig. 5.9: Illustration that more than basic morphological operations are needed on intersection map of edge-based orthogonal scan to obtain accurate object mask. (a),(b) Results of horizontal and vertical scan, respectively. (c) The intersection map.

### 5.3.1 Illustration of Algorithm Steps

Consider segmenting frame 74 of the Mother and Daughter sequence. Fig. 5.10 illustrates the working of motion-related analysis. Fig. 5.10(a) and (b) show the original frames 73 and 74. Fig. 5.10(c) shows the foreground object  $O_{1,73}$  as already segmented from frame 73. Fig. 5.10(d) shows the change detection result  $CD_{74}$  obtained from analyzing frames 74 and 71 (i.e., we have let  $d = 3$ ). We employ the three-step motion estimation algorithm for the boundary blocks. Because the sequence is of the videophone type, the motion search range is  $\pm 7$  pixels. Fig. 5.10(e) shows the union of  $P_{1,74}$  and  $CD_{74}$ , an intermediate result in forward tracking. And Fig. 5.10(f) shows the coarse mask  $PCD_{1,74}^B$  after backward validation.

Fig. 5.11 illustrates the working of the mask refinement procedure. Fig. 5.11(a) shows an intermediate result of mask rounding, namely, the union mask of the two orthogonal scan results. The final result of mask rounding, i.e.,  $MR_{1,74}$ , after erosion of invalid outer pixels is shown in Fig. 5.11(b). Fig. 5.11(c) and (d) show the row and the column maps, respectively, obtained in the first step of *footprint tightening*. Note the horizontal white stripes that protrude into the mother's and the daughter's head areas in Fig. 5.11(c), and

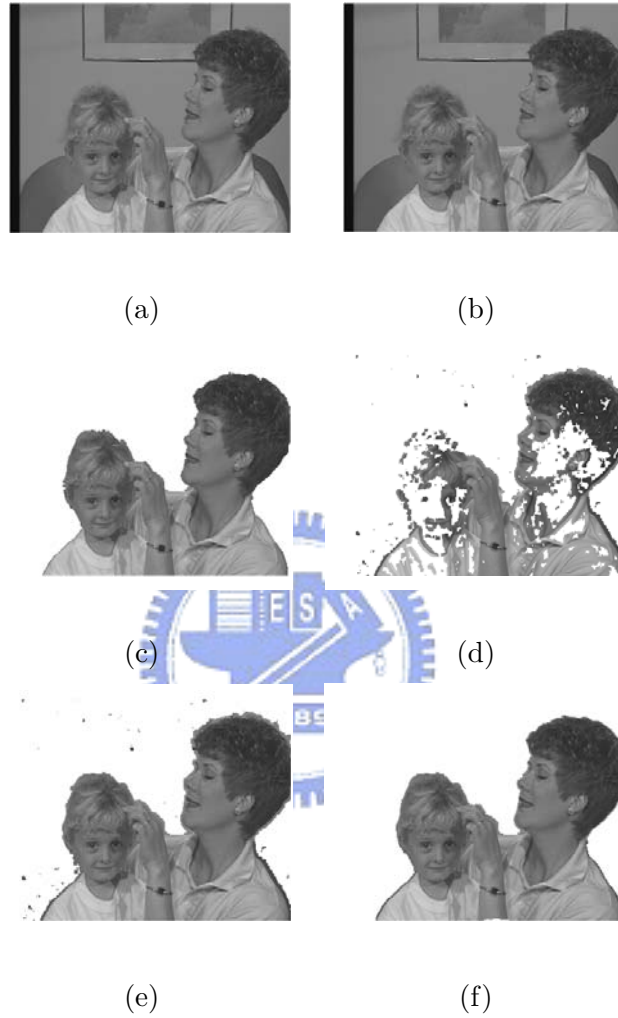


Fig. 5.10: Illustration of the mother-related analysis in the proposed algorithm using frame 74 of the Mother and Daughter sequence as example. (a),(b) Original frame 73 and 74, respectively. (c) Foreground object  $O_{1,73}$  segmented from frame 73. (d) Change detection result  $CD_{74}$ . (e) Union of  $P_{1,74}$  and  $CD_{74}$ . (f) Coarse mask  $PCD_{1,74}^B$ .

Table 5.1: Number of Pixels Deleted in Each Iteration of Boundary Sharpening for the First Frame

Iteration No.	1		2		3	
Step	Pruning	Filling	Pruning	Filling	Pruning	Filling
Akiyo	1724	501	4	3	3	1
Claire	1037	827	10	3	0	0
Mother-Daughter	1637	448	7	3	1	0
Salesman	1430	257	8	5	2	0
Foreman	1613	701	10	7	2	0

the vertical white stripes that protrude into the mother's and the daughter's hair areas in Fig. 5.11(d). They indicate the existence of gaps in the boundary edges found using the Canny edge detector. Fig. 5.11(e) shows the final footprint-tightened mask,  $FT_{1,74}$ . In this case, it is only slightly different from the mask rounding result. Fig. 5.11(f) and (g) show the results of boundary sharpening in the horizontal and the vertical directions, respectively, each for one iteration. Note the difference between the front contours for the mother's face. There are other small but noticeable differences between the object boundaries in Fig. 5.11(f) and (g), too. Fig. 5.11(h) shows the intersection map of Fig. 5.11(f) and (g).

We mentioned that the steps in boundary sharpening can be iterated until convergence. Tab. 5.1 gives the convergence behavior in processing the first frame of some common CIF test sequences. The convergence behavior for later frames is substantially similar, only that a lower portion of pixels are involved. We thus see that, as mentioned, the work is largely finished in the first iteration. Hence one iteration should suffice.

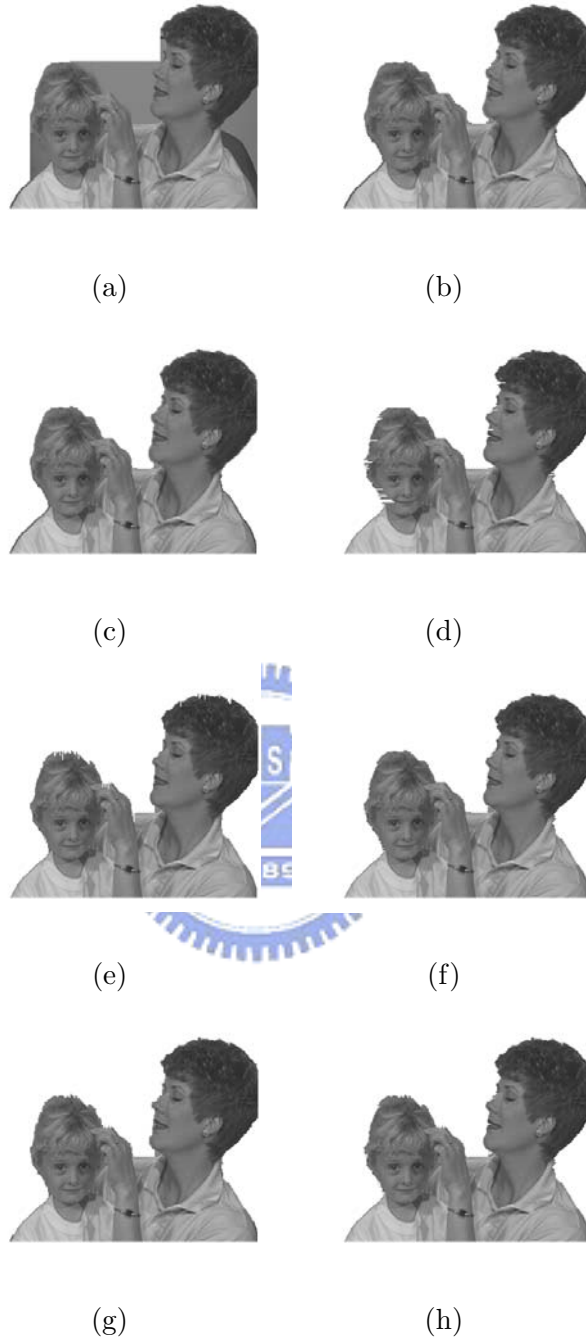


Fig. 5.11: Illustration of the mask refinement procedure in the proposed algorithm using frame 74 of the Mother and Daughter sequence as example. (a) Union mask of two orthogonal scan results. (b) Result of mask rounding,  $FG_r$ . (c),(d) Row and column maps obtained in footprint tightening, respectively. (e) Result of footprint tightening,  $FG_t$ . (f),(g) After one iteration of boundary sharpening in horizontal and vertical directions, respectively. (h) Intersection map of  $FG_s$ .

### 5.3.2 Algorithm Performance

Fig. 5.12- 5.15 present some segmentation results for several common CIF test sequences of different levels of segmentation difficulty. Fig. 5.12 shows the results for the Akiyo sequence, one of the easier due in part to the smooth shape of the moving object (the person). Fig. 5.13 shows some results for the Mother and Daughter sequence, which is somewhat more difficult than Akiyo due to the motion content and the non-convex shape of the moving foreground (the two persons). In both cases, we see that the object boundaries are subjectively quite accurately identified. Fig. 5.14 shows some results for the Salesman sequence. This sequence is a challenge to video segmentation algorithms due in part to the high background noise and the moving shadow in the front. We see that the segmentation result, though still not completely ideal, delineates the moving person rather accurately. Finally, Fig. 5.15 shows some results for the Foreman sequence. With a violently moving background, this sequence is probably the most challenging of the four. In addition, the low contrast between the foreman's helmet and the building behind makes their separation difficult based on change detection. In fact, to the authors' knowledge, no published automatic video segmentation algorithms to date seem to demonstrate good segmentation performance for this sequence. Towards a better segmentation, one possibility is to incorporate region-based analysis (such as the technique presented in [22] or watershed-type analysis). Another possibility is to invoke background (or global) motion estimation. These and other possible remedies are left to potential future work.

Besides subjective evaluation, objective evaluation of the segmentation performance is often also of interest. A reference segmentation mask for the Akiyo sequence is available. We thus also evaluate the segmentation accuracy using the measure proposed in [60], which calculates the fractional agreement between the segmented object mask and the reference



Fig. 5.12: Example segmentation results of the Akiyo sequence. Left, top to bottom: Original frame, 30, 70, and 117. Right: segmentation results.



Fig. 5.13: Example segmentation results of the Mother and Daughter sequence. Left, top to bottom: Original frame, 50, 95, and 120. Right: segmentation results.



Fig. 5.14: Example segmentation results of the Salesman sequence. Left, top to bottom: Original frame, 10, 20, and 30. Right: segmentation results.





Fig. 5.15: Example segmentation results of the Foreman sequence. Left, top to bottom: Original frame, 4, 24, and 34. Right: segmentation results.

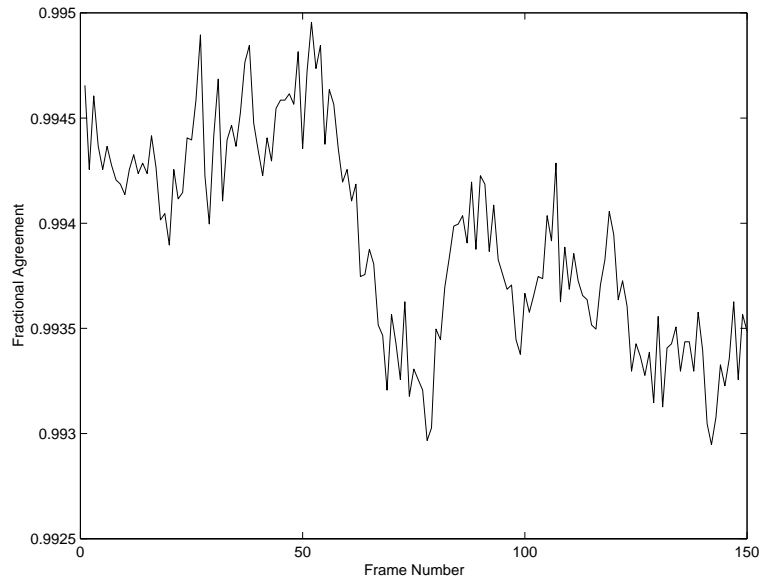


Fig. 5.16: Fractional agreement between the segmented foreground object of the Akiyo sequence and the reference mask.

mask for each frame. Fig. 5.16 shows the result. We see that the fractional agreement is over 0.993 nearly all the time. Further, Fig. 5.17 shows the temporal coherency of the segmentation measured by fractional agreement between the object masks obtained for two adjacent frames. We see that the coherency is close to that of the reference mask, sometimes higher.

### 5.3.3 Algorithm Speed

We did some high-level analysis of the algorithm complexity in the last section. There we noted that the precise amount of computation depended on detailed organization of the program and the video material. As a result, it is difficult to characterize exactly. We therefore present some computing time data for several common CIF test sequences. Tab. 5.2 lists the data from using a personal computer with 1.8-GHz Pentium IV CPU. The program is

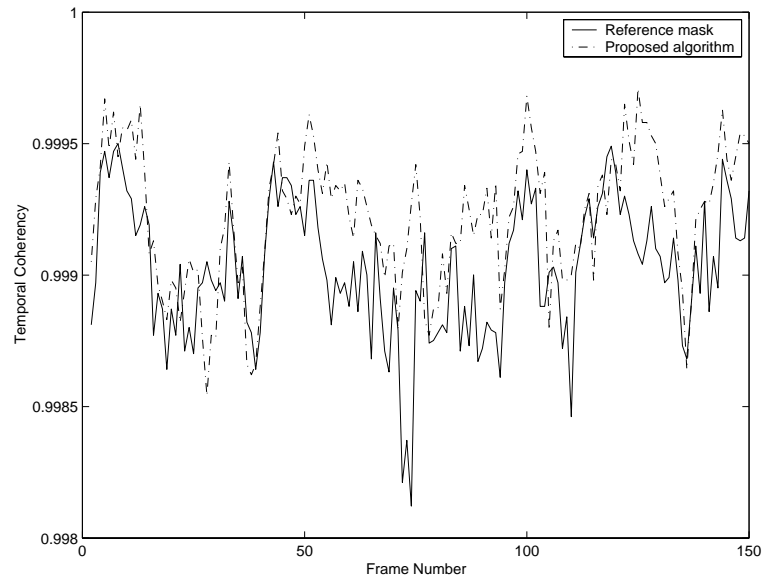


Fig. 5.17: Fractional agreement between the segmented foreground object of the Akiyo sequence in adjacent frame.

not optimized. With suitable optimization of the program, the speed should be significantly faster. Thus the algorithm is suitable for real-time desktop or portable multimedia and videoconferencing applications.

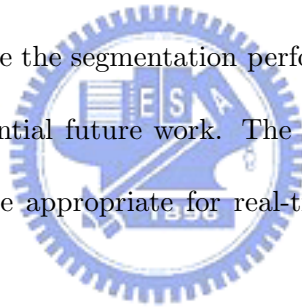
## 5.4 Discussion

We presented an algorithm for automatic segmentation of moving objects in natural video scenes. Novel edge-based morphological processing methods were designed to facilitate accurate determination of the object boundaries. The algorithm can handle grossly non-convex object shapes, which are commonplace in typical natural video. A novel, low-complexity motion estimation technique was also designed to aid robust object tracking. Experimental results showed that the algorithm achieved good performance, except for videos with vio-

Table 5.2: Average Computing Time in ms per Frame of Major Algorithm Functions

Algorithm Functions	Change Detection	Motion Estimation	Mask Refinement
Akiyo	7.63	11.6	10.7
Claire	7.61	11.9	11.3
Mother-Daughter	7.60	13.7	13.2
Salesman	7.81	10.6	9.1
Foreman	7.57	19.6	11.6

lently moving background and low contrast between the moving object and the background. Several possible ways to improve the segmentation performance for such videos were pointed out. And they are left to potential future work. The required computing time of the proposed algorithm was seen to be appropriate for real-time desktop or portable multimedia applications.



## Chapter 6

# Support of Content-Based Applications



This chapter describe techniques for authoring tools [6], and integrating the foreground objects and background images. Firstly, shrinkage and enlargement are two of the basic techniques. The complexity of the techniques are analyzed. Then we illustrate a technique fits video synthesis and superimposition to the video sequence.

In using the segmentation results in scene composition, we frequently need to enlarge or shrink the objects. We consider how these can be done efficiently with good performance, in a manner that is also suitable for spatial-scalable coding.

### 6.1 Methods for Object Enlargement and Shrinkage

Image interpolation is employed to add the resolution of an image by estimating the pixel intensities on an upsampled position. The most commonly used image interpolation methods, such as nearest neighbor interpolation, bicubic interpolation and bilinear interpolation, can

cause blocky interpolated images with jaggy edges. Several schemes for edge preserving interpolation have been presented. Jensen and Anastassion [24], estimate the local areas' edge positions and directions by a truncated Fourier expansion series when the shape of small patches of an image which have edges across them. The nonlinear operations can reconstruct high frequency components. Ting [52] presents an edge orientation-directed interpolation. The proposed edge preserving interpolation method consists of two stages. The first stage is the fuzzy logic system that realizes edge contrast preservation by adaptively synthesizing the interpolated pixel based on the local edge gradients. The second stage detects the significant edges by checking several structural elements and readjusts the interpolated edge pixels according to the detected edge orientation. Besides, Schultz and Stevenson [?] use a Bayesian approach to preserve edges and other discontinuities in image expansion. In addition, Ratakondo and Ahuja [45] present an iterative interpolation method, in which the image interpolation problem is formulated as a inverse problem.

Consider scene composition using the extracted video objects, enlarge or shrink the segmentation results are frequently needful tools. This is expected to require often enlargement or shrinkage of the video objects. For convenience, we consider employing the interpolation and decimation filters specified in MPEG-2 [20], which are  $[-12, 140, 140, -12]/256$  for interpolation and  $[-29, 0, 88, 138, 88, 0, -29]/256$  for decimation. The frequency responses of these filters are shown in Fig. 6.1.

Several modifications, such as considering edge direction to cope with jaggy edges [24, 52], are necessary. For simplicity, presented methods use MPEG-2 filters. MPEG-2 only considers enlargement and shrinkage by a factor of two (both horizontally and vertically), but we consider arbitrary-factor interpolation and decimation. And second, MPEG-2 only considers interpolation and decimation for rectangular video frames, whereas we have arbitrarily

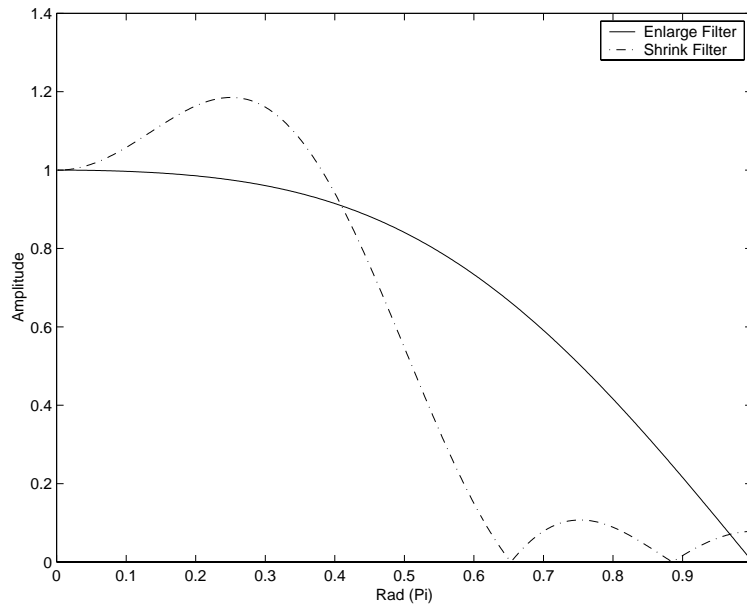


Fig. 6.1: Amplitude response of the MPEG-2 enlargement and shrinkage filters.

shaped video objects. Consider enlargement by arbitrary integer factors first. We consider a simple interpolation method illustrated by example in Fig. 6.2. In essence, enlargement by an integer-power-of-2 (say “2”) times is accomplished by repeated application of the MPEG-2 interpolation filter  $n$  times. Enlargement by a factor that is not an integer power of 2 is accomplished by enlarging to the nearest lower integer power of 2, followed by linear interpolation between two nearest pixels according to the relative pixel distances. Shrinkage by arbitrary integer factors-operates on a similar principle and is illustrated also by an example in Fig. 6.3. Again, shrinkage by an integer-power-of-2 times is accomplished by repeated application of the MPEG- 2 decimation filter. Shrinkage by a factor that is not an integer power of 2 is accomplished by shrinking to the nearest lower integer power of 2, followed by linear interpolation between two nearest pixels according to the relative pixel distances. A rational-factor (say,  $p/q$  where  $p$  and  $q$  are integers) enlargement or shrinkage can be obtained, in principle, by conducting an  $p$ -times interpolation and  $q$ -times decimation.

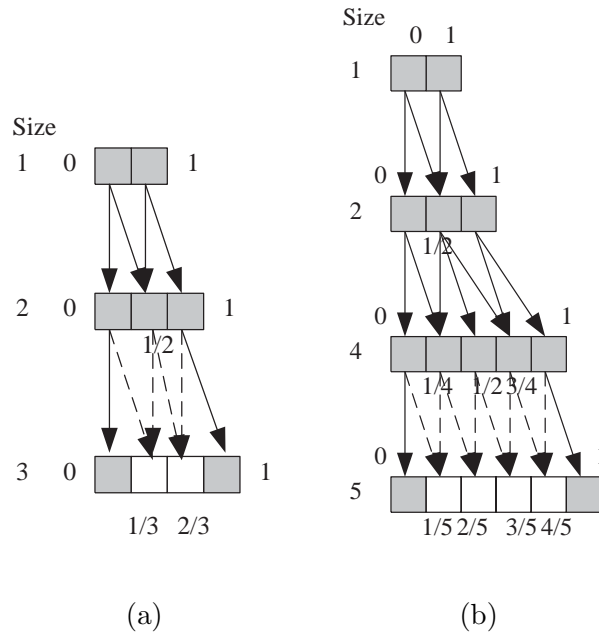


Fig. 6.2: Proposed enlargement algorithm, illustrated by example. (a) Enlargement by two or three times, (b) Enlargement by two, four, or five times (Gray squares denote pixels in the original object mask or as obtained by interpolation using the MPEG-2 filter; white squares denote linear linear interpolated pixels. Dashed lines indicate linear interpolation between nearest pixels.)

In either enlargement or shrinkage, when the filter memory extends beyond the object mask (this happens when operating on pixels near the object boundary), we repeat the boundary pixel values for filtering purpose, similar to the principle used in H.263 and MPEG-4 for motion estimation. Note also that the way object enlargement and shrinkage are conducted fits well into a context of spatial-scalable coding.



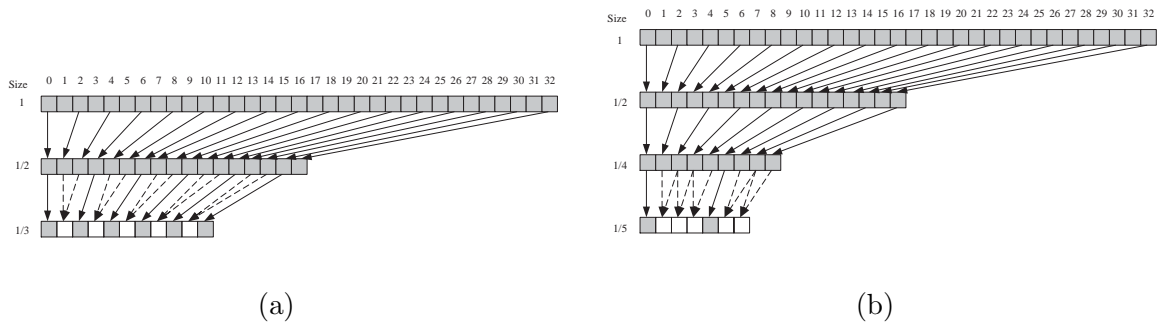
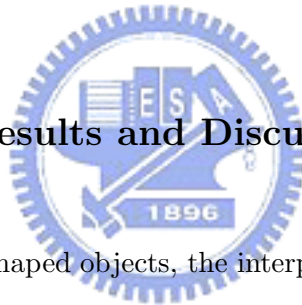


Fig. 6.3: Proposed Shrinkage algorithm, illustrated by example. (a) Shrinkage by two or three times, (b) Shrinkage by two, four, or five times (Gray squares denote pixels in the original object mask or as obtained by decimation using the MPEG-2 filter; white squares denote linear linear interpolation pixels. Dashed lines indicate linear interpolation between nearest pixels.)

## 6.2 Experimental Results and Discussions



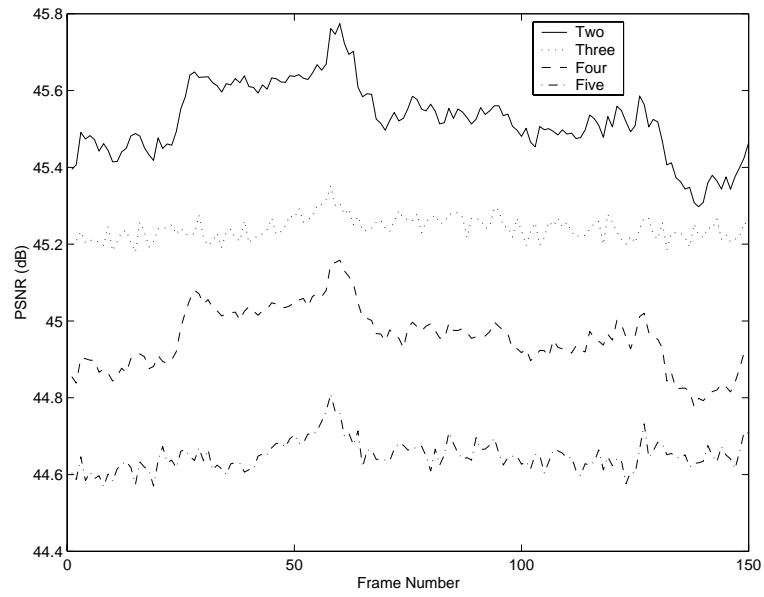
Since we deal with arbitrarily shaped objects, the interpolation and decimation performance at object boundaries is of particular interest. For this we discussed a way to enlarge or shrink arbitrarily shaped video objects. In Fig. 6.4 we show some results on the similarity (in PSNR) between the original segmented object and the resulting one that is enlarged  $X$  times and shrunk to the original size, where “ $X$ ” is between 2 and 5. As expected, the interior pixels (inside a three-pixel-wide band at object boundary) are subject to less distortion from enlargement and shrinkage compared to pixels near the object boundary. But overall, the PSNR is reasonably high.

Fig. 6.5 shows two synthesized scenes (two frames each) obtained by superimposing the moving objects segmented (and shrunk by two times) from the Claire, the Akiyo, and the Salesman sequences onto some background and middle ground, and with an additional fore-

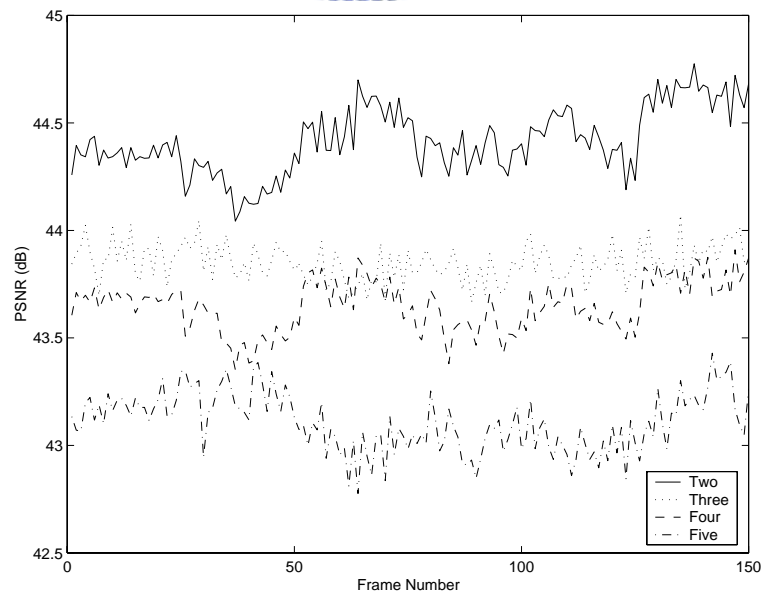
ground overlaid on them. The synthesized scenes appear natural.

Future work, considering Ting's methods [52], will employ edge preserving interpolation method consists for object enlargement. Moreover some applications are frequently necessary to present an object's motion details or layered syntheses. A motion line can describe the motion history of foregrounds. This is particularly useful tools in fields such as athletic and dance analyses ,or entertainment, such as Fig. 6.6(a). The result shows the superposed ball in the Table-tannis. The result presents the history of the ball track. Additionally, if a people want to exhibit a valued work of art, he can shoot the art in showroom and shoot the narrator's pre-exposition. The explained films can be composed of the two shots, such as Fig. 6.6(b). The result presents a fusion example from extracted VO in the Mother-and-Daughter and a statue.





(a)

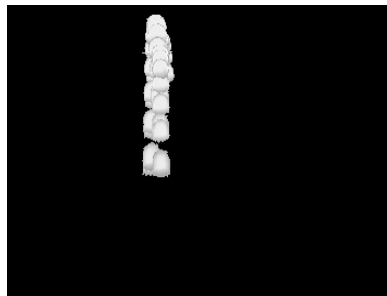


(b)

Fig. 6.4: Similarity, in PSNR, between the original segmented video object in CIF Mother-and-Daughter sequence and that enlarged  $X$  times and shrunk to original size, where  $X$  is between 2 and 5. (a) For interior of object (inside a three-pixel-wide band at object boundary). (b) For the three-pixel-wide band at object boundary.



Fig. 6.5: Synthesized scenes using segmented (and shrunk by 2 times) video objects and other contents.



(a)



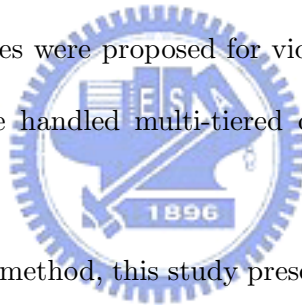
(b)

Fig. 6.6: Applications from superposed VOs. (a) Motion history of the ball in the Table-tennis. (b) VO in the Mother-and-Daughter fuse a statue for exhibition.

## Chapter 7

# Conclusions and Future Work

This thesis surveyed video segmentation methods and content-based applications of natural video sequences. Four techniques were proposed for video segmentation, one of which dealt with region-based method, one handled multi-tiered object structure, and the other two considered edge-linking design.



Regarding the region-based method, this study presented a method that featured designs to help facilitate good region segmentation while limiting the computational complexity. First, a preliminary seed-area identification and a final re-segmentation process are conducted on each video frame to help determine homogeneous areas and conduct region tracking. Then, a simple method of measuring homogeneity of texture in a region was devised and segmentation attempted to locate object boundaries where the texture exhibits significant changes. Next, a reduced-complexity motion estimation technique was used, so that dense motion fields can be calculated at a reasonable complexity. The proposed method could enhance segmentation performance.

In relation to multi-tiered object segmentation, this study presented an algorithm employing low-level spatio-temporal signal processing, and the segmentation was based on analysis

of edges, textural homogeneity of image regions, interframe pixel value changes, and apparent motion. Experiments involving several different kinds of video, including one difficult talking-head sequence (the Salesman) and one involving relatively fast motion (the Flower Garden) showed that the algorithm yielded reasonably good identification of object boundaries. While the algorithm was developed based on the assumption of a stationary background, in principle it can also address situations involving global background motion. The objective evaluation based on the Akiyo sequence shows that this method outperforms the region-based method, but has high computational complexity.

In terms of edge-linking methods, this study devised new algorithms for video object extraction. The algorithms employed edge analysis for accurately determining object boundaries. This work proposed a method based on the shortest path algorithm. Moreover, this study presented a morphological filtering algorithm based on adjacent pixels. The former method is more complicated than the latter. The error is below 0.7% in every frame, and generally is even better. Motion estimation achieves the greatest reduction in the latter, at 3.88 times that of full search. Compared to the existed edge-linking approaches in [27, 37], this study offered objective assessments of the performance change associated with the boundary correction. A characteristic of the edge-linking approach is that it is designed for use with still background videos, such as videoconference applications. For moving backgrounds, more effective approach of global motion detection is necessary.

For scene composition, this study developed simple object enlargement and shrinkage methods. When object sizes are suitable, objects overlapped with objects or images via multi-layer superimposition to establish new scenes. Additionally, for checking motion history, object overlaps and their transparencies provide a useful means of tracking the motion manners.

This thesis has shown the effectiveness of the proposed approaches to video segmentation. However, further work is required to enhance the algorithms, and a number of issues require further study:

1. The motion estimation algorithm influences processing speeds and enabling applications. To achieve real-time processing and handle shacked cameras, faster global motion estimation is required. When the global motion information has been identified, the local motion information can be further examined, along with locating foregrounds and still backgrounds. Tough questions include aperture problem and occlusion problems.
2. The statistical model of motion is considered to improve the usability, which may be integrated to the region-based frame work. For example, a Markov Random Fields (MRF) [62] approach can be pursued. This approach could particularly effective for segmenting scenes involving moving backgrounds, but must consider methods of reducig complexity.
3. Application development video segmentation is an enabling technology. This technology represents the basis of numerous video analysis technologies. The developed methods yield object results, and thus provide a basis for applications. For example, video description tools (authoring) could be developed, which consider a mosaic based on the background, and the manner in which the foreground object moves over this background and displays related shape changes.

# Bibliography

- [1] T. Aach, A. Kaup, and R. Mester, “Statistical model-based change detection in moving video,” *Signal Processing*, vol. 31, no. 2, pp. 165–180, Mar. 1993.
- [2] F. E. Alsaqre and B. Yuan, “Moving object segmentation for video surveillance and conferencing applications,” in *Int. Conf. Commun. Technol. Proc.*, vol. 2, Apr. 2003, pp. 1856–1859.
- [3] M. Bierling, “Displacement estimation by hierarchical blockmatching,” *SPIE* vol. 1001, *Visual Commun. Image Processing*, 1988, pp. 387–403.
- [4] G. D. Borshukov, G. Bozdagi, Y. Altunbasak, and M. Tekalp, “Motion segmentation by multistage affine classification,” *IEEE Trans. Image Processing*, vol. 6, no. 11, pp. 1591–1997, Nov. 1997.
- [5] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679–698, Nov. 1986.
- [6] K. Cha and S. Kim, “MPEG-4 studio: An object-based authoring system for MPEG-4 contents,” *J. Multimedia Tools Applications*, vol. 25, no. 1, pp. 111–131, Jan. 2005.



- [7] Y.-J. Chang, C.-C. Chen, J.-C. Chou, and Y.-C. Chen, "Implementation of a visual chat room for multimedia communications," in *IEEE Workshop Multimedia Signal Processing*, 1999, pp. 599–604.
- [8] H.-J. Chen and Y. Shirai, "Segmentation based on accumulative observation of apparent motion in long image sequences," *IEICE Trans. Inf. & Syst.*, vol. E77-D, no. 6, pp. 694–704, June 1994.
- [9] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 577–586, July 2002.
- [10] S.-Y. Chien, Y.-W. Huang, and L.-G. Chen, "Predictive watershed: a fast watershed algorithm for video segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 5, pp. 453–461, May 2003.
- [11] J. G. Choi, S.-W. Lee, and S.-D. Kim, "Spatio-temporal video segmentation using a joint similarity measure," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 279–286, Apr. 1997.
- [12] P. Daras, I. Kompatsiaris, I. Grinias, G. Akrivas, G. Tziritas, S. Kollias and M. G. Strintzis, "MPEG-4 authoring tool using moving object segmentation and tracking in video shots," *EURASIP J. Applied Signal Processing*, vol. 2003, no. 9, pp. 861–877, Aug. 2003.
- [13] D. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.

- [14] M. Flickner *et al.*, “Query by image and video content: The QBIC system,” *in IEEE Computer*, vol. 28, no. 9, pp. 23–32, Sep. 1995.
- [15] A. C. M. Fong, S. C. Hui, M. and K. H. Leung, “Content-based video sequence interpretation,” *IEEE Trans. Consumer Electronics*, vol. 47, no. 4, pp. 873–879, Nov. 2001.
- [16] H. Gao, W.-C. Sju, and C.-H. Hou, “Improved techniques for automatic image segmentation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 12, pp. 1273–1280, Dec. 2001.
- [17] D. Gatica-Perez, C. Gu, and M.-T. Sun, “Semantic video object extraction using four-band watershed and partition lattice operators,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, pp. 603–618, May 2001.
- [18] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley, 1992.
- [19] H. Gu, Y. Shirai, and M. Asada, “Motion description and segmentation of multiple moving objects in a long image sequence,” *IEICE Trans. Inf. and Syst.*, vol. E78-D, no. 3, pp. 277–289, Mar. 1995.
- [20] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital video: An introduction to MPEG-2.*, Springer, 1996.
- [21] Y.-H. Jan and D. W. Lin, “Extraction of video objects by combined motion and edge analysis,” *in Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, May 2000, pp. 677–680.
- [22] Y.-H. Jan and D. W. Lin, “Video Segmentation with extraction of overlaid objects via multi-tier spatio-temporal analysis,” *Int. J. Electrical Engineering.*, vol. 11, no. 3, pp. 205–218, Aug. 2004.

- [23] Y.-H. Jan and D. W. Lin, "Automatic video segmentation with a novel edge-based morphological filtering," in *Proc. IEEE Int. Symp. Consumer Electronics.*, Reading, UK, 2004, pp. 340–344, Sep.
- [24] K. Jensen and D. Anastassion, "Subpixel edge localization and interpolation of still images," *IEEE Trans. Image Processing*, vol. 4, pp. 285–295, Mar. 1995.
- [25] M. Kim and J. Kim, "Moving video object segmentation using statistical hypothesis testing," *Electron. Lett.*, vol. 36, no. 2, pp. 128–129, Jan. 2000.
- [26] M. Kim, J. G. Choi, D. Kim, H. Lee, M. H. Lee, C. Ahn, and Y. S. Ho, "A VOP Generation tool: Automatic segmentation of moving objects in image sequences based on spatio-temporal information," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 8, pp. 1216–1226, Dec. 1999.
- [27] C. Kim and J. N. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 2, pp. 122–129, Feb. 2002.
- [28] I. Kompatsiaris and M. G. Strintzis, "Spatiotemporal segmentation and tracking of objects for visualization of videoconference image sequences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 8, pp. 1388–1402, Dec. 2000.
- [29] K. W. Lee and J. Kim, "Moving object segmentation based on statistical motion model," *Electron. Lett.*, vol. 35, no. 20, pp. 1719–1720, April 1999.
- [30] M. E. Lukacs and D. G. Boyer, "A universal broadband multipoint teleconferencing service for the 21st century," *IEEE Commun. Mag.*, vol. 33, no. 11, pp. 36–43, Nov 1995.

- [31] H. Luo and A. Eleftheriadis, “Rubberband: an improved graph search algorithm for interactive object segmentation,” in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, 2002, pp. 101–104.
- [32] H. Luo, A. Eleftheriadis, “Model-based segmentation and tracking of head-and-shoulder video objects for real time multimedia services,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 3, pp. 379–389, Sep. 2003.
- [33] A.-R. Mansouri and J. Konrad, “Multiple motion segmentation with level sets,” *IEEE Trans. Image Processing*, vol. 12, no. 2, pp. 201–220, Feb. 2003.
- [34] F. Marques and C. Molina, “An object tracking technique for content-based functionalities,” in *SPIE vol. 3024, Visual Commun. Image Processing*, 1997, pp. 190–198.
- [35] T. Meier and K. N. Ngan, “Automatic segmentation of moving objects for video object plane generation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 525–538, Sep. 1998.
- [36] T. Meier and K. N. Ngan, “Segmentation and tracking of moving objects for content-based video coding,” *IEE Proc.-Vis. Image Signal Process.*, vol. 146, no. 3, pp. 144–150, June 1999.
- [37] T. Meier and K. N. Ngan, “Video segmentation for content-based coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 8, pp. 1190–1203, Dec. 1999.
- [38] A. Neri, S. Colonnese, G. Russo, and P. Talone. “Automatic moving object and background separation,” *Signal Processing*, vol. 66, no. 2, pp. 219–232, Apr. 1998.

- [39] W. Niblack *et al.*, “The QBIC project: Querying images by content using color, texture and shape,” in *IS&T/SPIE, Storage and Retrieval for Image and Video Databases*, Feb. 1993, pp. 173–187.
- [40] E. P. Ong, B. J. Tye, W. S. Lin, and M. Etoh, “An efficient video object segmentation scheme,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, vol. 4, 2002, pp. 3361–3364.
- [41] N. Otsu, “A threshold selection method from gray-level histogram,” *IEEE Trans. System Man Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [42] I. Patras, E. A. Hendriks, and R. L. Lagendijk, “Video segmentation by MAP labeling of watershed segments,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 3, pp. 326–332, Mar. 2001.
- [43] A. P. Pentland *et al.*, “Photobook: tools for content-based manipulation of image database,” in *IS&T/SPIE, Storage and Retrieval for Image and Video Database II*, Feb. 1994, pp. 34–47.
- [44] J. Pons, J. Prades-Nebot, A. Albiol, and J. Molina, “Fast motion detection in compressed domain for video surveillance,” *Electron. Lett.*, vol. 38, no. 9, pp. 409–411, April 2002.
- [45] K. Ratakondo and N. Ahuja, “POCS based adaptive image magnification,” in *Proc. IEEE Int. Conf. Image Processing*, 1998, pp. 203–207.
- [46] P. Salembier, P. Brigger, J. R. Casas, and M. Padas, “Morphological operators for image and video compression.” *IEEE Trans. Image Processing*, vol. 5, no. 6, pp. 881–898, June 1996.

- [47] P. Salembier, F. Marques, M. Pardas, R. Morros, I. Corset, S. Jeannin, L. Bouchard, F. Meywe, and B. Marcotegui, "Segmentation-based video coding system allowing the manipulation of objects." *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 60–74, Feb. 1997.
- [48] Y. Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering*. Boca Raton, Florida: CRC Press, 2000.
- [49] P. M. Spira and A. Pan, "On finding and updating spanning trees and shortest paths," *SIAM J. Computing*, vol. 4, no. 3, pp. 375–380, 1975.
- [50] S. Sun, D. R. Haynor, and Y. Kin, "Semiautomatic video object segmentation using VSnares," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 1, pp. 75–82, Jan. 2003.
- [51] A. M. Tekalp, *Digital Video Processing*. Upper Saddle River, New Jersey: Prentice Hall, 1995.
- [52] Hou-Chun Ting, "Edge orientation-directed interpolation of digital images and its applications," Ph.D. dissertation, Dept. of Electronics Engineering, National Chiao Tung University, 1997.
- [53] M. Thorup, "Undirected single-source shortest paths with positive integer weights in linear time," *J. ACM*, vol. 46, no. 3, pp. 362–394. May 1999.
- [54] Y. Tsaig and A. Averbuch, "Automatic segmentation of moving objects in video sequences: a region labeling approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 597–612, July 2002.

- [55] E. Tuncel and L. Onural, "Utilization of the recursive shortest spanning tree algorithm for video-object segmentation by 2-D affine motion modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10. no. 5, pp. 776–781, Aug. 2000.
- [56] J. Y. A. Wang and E. H. Adelson, "Spatio-temporal segmentation of video data." in *SPIE vol. 2182, Image and Video Proceeding II*, Feb. 1994, pp. 625–638.
- [57] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Trans. Image Processing*, vol. 3. no. 5, pp. 625–638. Sep. 1994.
- [58] D. Wang, "Unsupervised video segmentation based on watershed and temporal tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 539–546, Sep. 1998.
- [59] S. Weik, J. Wingbermuehle, and W. Niem, "Automatic creation of flexible antropomorphic models for 3D videoconferencing," in *Proc. Computer Graphics*, 1998, pp. 520–527,
- [60] M. Wollborn and R. Mech, "Refined procedure for objective evaluation of VOP generation algorithms," Doc. ISO/IEC JTC1/SC29/WG11 MPEG98/3448, Mar. 1998.
- [61] C. S. Won, "A block-based MAP segmentation for image compression." *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8. no. 5, pp. 592–601, Sept. 1998.
- [62] W. Zeng and W. Gao, "Semantic object segmentation by a spatio-temporal MRF model," in *Proc. IEEE Int. Conf. Pattern Recognition*, vol. 4, pp. 775–778, pp. 1021–1025, Aug. 2004.