Object Tracking with Dynamic Feature Modeling and Frame Watermarking via Embedded Filters

BY

ZHUAN QING HUANG

A THESIS SUBMITTED TO UNIVERSITY OF WESTERN SYDNEY FOR THE DEGREE OF PHILOSOPHY (PH.D)
FEBRUARY 2009
Declaration

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. No part of the work done here has ever been submitted in support of an application for another degree of this or any other university or other institute of learning.

Zhuan Qing Huang
Acknowledgements

I would like to thank many people who have supported me over the years in pursuing my academic goals. I am greatly indebted to my supervisor, Dr Zhuhan Jiang, for his continuous encouragement, advice, help and invaluable suggestions. His guidance and inspiration are essential in completing my PhD.

Special thanks to my co-supervisor, Professor Yan Zhang, for his continuous support for many years, and providing me an excellent research environment as well as additional funds during my study. I would also like to thank the members in the Intelligent Systems Laboratory, Yun Bai, Dongmo Zhang, and others. I have shared a great time with them and get inspiration from their research experience.

I would like to thank the University of Western Sydney and the Australia Commonwealth Fund for the Australian Postgraduate Award scholarship that made my academic pursuits possible.

I would like to thank my family for their continual support.
Abstract

This thesis investigates the object tracking problem based on modeling the object features in a video sequence in terms of distinctive color characteristics, or a hybrid along with spatial correlation and motion properties, and it also studies the related ownership protection for the individual video frames, even potentially the object itself, through watermarking via embedded wavelet filters.

Tracking an object in a video sequence poses a significant challenge in its wide applications including process control, medical diagnosis, human computer interface, to name a few. The difficulty of the problem is often directly related to complexity of the object-occupied scene and its variation within the video, and is also dependent upon the critical feature modeling that underpins the tracking process. We propose in this thesis to model the distinctiveness of the non-rigid object within a moving background through identifying and analyzing discriminating colors, stable appearance, spatial relevance, and motion data, and to develop the corresponding methods to track objects in different type of scenes including the camouflaged object of interest.

Many existing tracking methodologies are based on plain object templates, histograms or detectable object contours. Such a template approach is often susceptible to changes in object shape, illumination, local appearance, or to partial object occlusion. Some approaches considered certain local appearance changes but
are limited to knowing *a priori* how the changes take place, while others may have to resort to complicated data training in a long course to retrieve a stable feature. We thus propose a kernel based object model that transforms the object appearance into a statistical representation in term of grouped color probability densities. Quite different from the histogram approaches, this model represents the characteristics of the groups of pixels with implicit or explicit incorporation of their spatial information. We differentiate the importance on the selected groups of pixels by identifying distinctive features in term of stable appearance, the undesirable region of potentially large local appearance change, or areas of heavy object deformation. This framework provides the flexibility to model the object based on the dynamic characteristics of the object in shape and in the distribution of color groups. It is designed to effectively handle the deformable object, local changes or partial occlusion in the moving background, and can also greatly reduce the computational complexity by making smaller number of statistical samples.

We further model the distinctive features by means of the color contrast between the object and its background. We extract sections of color distribution density for the object that stand out distinctively from its local background, and use these to locate the object in the newer frames through the Bayesian estimation. We then propose to extract dominant elements for the distinctive features by maximizing the difference of the object and its local background through the optimal segmentation. The object is located in the newer frames via the pixel similarity to the extracted dominant elements. In contrast to the traditional approach in which only certain specific types of objects such as a bright target are applicable, or the feature selection is based directly
on the total difference of the densities, our proposed approach is generic and efficient with its dynamic and automatic extraction of updated distinctive feature in the process, and such features are explored through different color spaces or their derived properties.

One of the most challenging tracking problems is when object color and texture resemble that of the background, and when the object shape and the background also change in a video sequence. Most existing tracking algorithms will fail under such harsh environment or choose to completely stay away. In this regard, however, we propose an iterative method of Weighted Region Consolidation to track a camouflaged object. We will detect the object motion based on both spatial and intensity densities by locating pixels with high motion probabilities from the difference data of successive frames. We then consolidate the object region by weighted overall neighborhood intensity, and by a contour verification or voting method.

In the realm of videos or frame images, watermarks may be inserted for various reasons, including copyright protection on parts of the multimedia data or even embedding sporadic object cues for future searching. We propose to watermark frame images by encoding the watermark bits into the choice of the wavelet filters, in complete contrast to the watermarking convention. Such filters are selected from different classes in such a way that they lead to sufficiently distinguishing subbands upon different sequence of filters. This proposed scheme is scalable in that the methodology can be utilized to build a larger or a smaller system, and the scheme is also shown to be robust to noise injection, illumination changes and some forms of geometric distortion or cropping.
# Contents

1 INTRODUCTION................................................................................................... 1

1.1 Object detection and tracking ................................................................. 2

1.1.1 Object detection.................................................................................... 3

1.1.2 Object tracking ..................................................................................... 8

1.1.2.1 Template match ..................................................................... 9

1.1.2.2 Shape based approach ......................................................... 10

1.1.2.3 Colour based approach ......................................................... 16

1.1.2.4 Motion integration................................................................ 19

1.2 Watermarking .......................................................................................... 20

1.2.1 Spatial domain ...................................................................................... 22

1.2.2 Transform domain ............................................................................... 23

1.2.3 Video watermarking ............................................................................. 29

1.3 Challenges.............................................................................................. 33

1.4 Research objectives.................................................................................. 35

1.5 Contribution ............................................................................................ 37

1.6 Organization of the thesis ....................................................................... 41

2 KERNEL-BASED FEATURE MODEL ............................................................... 42

2.1 Introduction ............................................................................................... 42

2.2 Nonparametric density estimation ......................................................... 46

2.3 Spatial integration color model................................................................. 49

2.3.1 Feature modeling in geometry............................................................ 50

2.3.2 Measurement metric ........................................................................ 56

2.3.3 Shape verification in the outer region................................................. 58

2.3.4 Model parameters and their impact on performance.......................... 61

2.4 Adaptive kernel template ........................................................................ 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.1</td>
<td>Motion estimation</td>
<td>64</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Adapted elements</td>
<td>68</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Deformable object</td>
<td>69</td>
</tr>
<tr>
<td>2.5</td>
<td>Kernel-based pixel modeling</td>
<td>70</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Scene modeling</td>
<td>71</td>
</tr>
<tr>
<td>2.5.1.1</td>
<td>Color modeling</td>
<td>72</td>
</tr>
<tr>
<td>2.5.1.2</td>
<td>Spatial modeling</td>
<td>73</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Object extraction</td>
<td>76</td>
</tr>
<tr>
<td>2.6</td>
<td>Implementation and Experiments</td>
<td>79</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Experiments on spatial integration color model</td>
<td>80</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Experiments on adaptive kernel template</td>
<td>84</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Experiments on kernel based pixel model</td>
<td>89</td>
</tr>
<tr>
<td>2.7</td>
<td>Summary</td>
<td>93</td>
</tr>
<tr>
<td>3</td>
<td>BAYESIAN PROPAGATION WITH DISCRIMINATIVE OBJECT DENSITY</td>
<td>95</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>95</td>
</tr>
<tr>
<td>3.2</td>
<td>Choice of Features</td>
<td>98</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Differentiating features</td>
<td>98</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Desirable feature densities and their potential complements</td>
<td>101</td>
</tr>
<tr>
<td>3.3</td>
<td>Bayesian Propagation of Densities</td>
<td>104</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Density update via the Bayesian rule</td>
<td>104</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Motion features</td>
<td>107</td>
</tr>
<tr>
<td>3.4</td>
<td>Region-wise update</td>
<td>108</td>
</tr>
<tr>
<td>3.5</td>
<td>Implementation and analysis</td>
<td>110</td>
</tr>
<tr>
<td>3.6</td>
<td>Summary</td>
<td>118</td>
</tr>
<tr>
<td>4</td>
<td>DYNAMIC DOMINANT FEATURE ELEMENTS</td>
<td>120</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>130</td>
</tr>
<tr>
<td>4.2</td>
<td>Dynamic dominant elements</td>
<td>122</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Band selection</td>
<td>122</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Extraction of dominant elements</td>
<td>126</td>
</tr>
<tr>
<td>4.3</td>
<td>Similarity Rendering</td>
<td>129</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Similarity function</td>
<td>129</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Object extraction</td>
<td>131</td>
</tr>
</tbody>
</table>
4.4 Analysis and Implementation ................................................................. 134
4.5 Summary .............................................................................................. 143

5 CAMOUFLAGED OBJECT ......................................................................... 144

5.1 Introduction .......................................................................................... 144
5.2 Tracking camouflaged object ................................................................. 147
  5.2.1 The case of challenges .................................................................. 147
  5.2.2 Background synchronization ....................................................... 149
  5.2.3 Weighted region consolidation ..................................................... 151
    5.2.3.1 Motion region extraction .................................................... 151
    5.2.3.2 Threshold ........................................................................ 152
    5.2.3.3 Consolidation window size ............................................... 152
    5.2.3.4 Iterative processing .......................................................... 153
  5.2.4 Refine object region ...................................................................... 154
  5.2.5 Shape Integration ......................................................................... 155
  5.2.6 Contour evolution ........................................................................ 156

5.3 Contour voting ..................................................................................... 158
  5.3.1 Voting confidence ......................................................................... 159
  5.3.2 Confidence assignment ................................................................. 163
    5.3.2.1 Motion and object detection ................................................ 164
    5.3.2.2 Active contour .................................................................. 165
    5.3.2.3 Edge-based contour ........................................................... 166
  5.3.3 Voting evaluation .......................................................................... 166

5.4 Implementation and Demonstration .................................................... 168
  5.4.1 Camouflaged object ....................................................................... 168
  5.4.2 Contour evaluation ....................................................................... 165

5.5 Summary ................................................................................................ 175

6 WAVELET BASED IMAGE WATERMARKING ...................................... 176

6.1 Introduction .......................................................................................... 176
6.2 Wavelet system .................................................................................... 178
  6.2.1 Wavelet function ....................................................................... 178
  6.2.2 Wavelet filter construction ......................................................... 180
    6.2.2.1 Orthogonal wavelet ......................................................... 180
List of Figures

Figure 2.1 Estimation by a balloon estimator .............................................................. 48
Figure 2.2 Estimation by a sample point estimator ...................................................... 49
Figure 2.3 (a) Annular regions $S_k$. (b) Region-wise color density on $k$.............. 53
Figure 2.4 Pixel-wise verification ............................................................................. 60
Figure 2.5 Neighborhood mask ................................................................................. 60
Figure 2.6 (a) Object in previous frame. (b) The candidate region in the current frame. ........................................................................................................... 65
Figure 2.7 (a) Object structured kernel elements. (b) Search similar structured kernel elements in the current frame................................................. 68
Figure 2.8 Geometry meaning of $\sigma$ ................................................................... 75
Figure 2.9 (a) Spatial probabilities after scaling. (b) Before scaling. (c) Original image........................................................................................................... 76
Figure 2.10 (a) Original object. (b) Locating object. (c) Outer region verification. (d) Full object extracted ................................................................. 82
Figure 2.11 Object located with refined contour in sequences ................................. 82
Figure 2.12 Comparison on locating the object with two different similarity measures........................................................................................................ 83
Figure 2.13 Parrot movement and the traces of its centre ......................................... 83
Figure 2.14 (a) Blocks. (b) Locations of kernel elements. (c) Object location ........ 85
Figure 2.15 Kernel elements. (a) Blocks of 7x7 pixels. (b) Blocks of 11x11 pixels. (c) Blocks of 13x13 pixels ................................................................. 85
Figure 2.16 (a) Distribution of the kernel blocks. (b) Corresponding elements ....... 86
Figure 2.17 (a) 1st frame. (b) 85th frame. (c) Trace of object centers in the sequence ........................................................................................................... 87
Figure 2.18 Movement of object center ..................................................................... 88
Figure 2.19 Trace of object centers .......................................................................... 88
Figure 2.20 Object and background intensity density .............................................. 90
Figure 2.21 Posterior probability. ............................................................................. 90
Figure 2.22  (a) Probabilities with spatial model (single Gaussian). (b) Probability with varying standard variation for Gaussian function. (c) Probability with varying Gaussian and two simulated basic shapes. ................................. 91

Figure 2.23 (a) Locate object with better spatial model. (b) Locate object with constrained probabilities. ........................................................................................................... 91

Figure 2.24  (a) The motion of object center. (b) Contours indicating movement. ... 92

Figure 3.1 (a) Local region. (b) Object density, background density and mixture density. (c) Difference of densities $\Delta(I) = p_o(I) - p_b(I)$. (c) The ratio of densities $p_o(I)/p_b(I)$................................................................. 100

Figure 3.2 (a) Hue difference. (b) Saturation difference. (c) Intensity difference. ..... 99

Figure 3.3 (a) Object and background intensity density. (b) Densities difference. (c) Smoothness densities.......................................................................................... 103

Figure 3.4 Shape adjustment ..................................................................................... 110

Figure 3.5 Test video sequences. (a) Penguin. (b) Meerkat. (c) Bird. (d) Car......... 111

Figure 3.6 (a) Hue density. (b) Saturation density. (c) Intensity density. (d) Hue difference. (e) Saturation difference. (f) Intensity difference. (g), (h), (i) Object probability from hue, saturation, intensity respectively. .......... 111

Figure 3.7 (a) Hue density. (b) Saturation density. (c) Intensity density. (d), (e), (f) Object probability from hue, saturation and intensity respectively. .......... 112

Figure 3.8 (a) Hue density. (b) Saturation density. (c) Intensity density. (d) Object probability by hue. (e) Object probability by Saturation. (f) Object probability by Intensity. (g) Object probability by hue and saturation (h) Object probability by hue and intensity. (i) Object probability by saturation and intensity. .................................................................................. 113

Figure 3.9 (a) Intensity densities from object and background. (b) Object pixel probability from intensity. (c) Smoothness density. (d) Object probability from smoothness. (e) Object pixel probability from intensity and smoothness................................................................. 114

Figure 3.10 (a) Intensity densities. (b) Object pixel probability. (c) Static and motion component densities. (d) Object pixel probability from (c). .......... 115

Figure 3.11 Penguin sequence 1................................................................................. 116

Figure 3.12 Penguin sequence 2................................................................................. 117

Figure 3.13 Car sequences .......................................................................................... 117

Figure 3.14 Traces of the moving objects..................................................................... 118

Figure 4.1 Progression flow of the matching ............................................................ 121

Figure 4.2 (a) A color segmentation. (b) Typical local neighborhoods along the boundary. (b) Segmentation patterns in a sample circle.................................. 125

Figure 4.3 Object modeling....................................................................................... 135
Figure 4.4 Matching process ..................................................................................... 136
Figure 4.5 (a) First frame. (b) Second frame. (c) Object on its own .................... 137
Figure 4.6 (a) Total segmental disparity against band width Radius= 2,3,4,5. (b) Radius=5 ................................................................. 137
Figure 4.7 Segmentation with w=5,15,25,35,45,55. .............................................. 138
Figure 4.8 (a) Bands chosen for the green color. (b) Segmentation with bands in (a). (c) Bands chosen for the red color. (d) Segmentation with bands in (c) .............................................................................................................. 139
Figure 4.9 Probabilities by elements ....................................................................... 140
Figure 4.10 (a) Total probabilistic from the selected bands. (b) Probabilistic after filtering. (c) Applied a threshold on the candidate region. (c) Projected shape. (e) Shape position after adjustment. (f) The refined object .............................................................................................................. 140
Figure 5.1 Video sequence ....................................................................................... 148
Figure 5.2 Contour extraction ................................................................................... 158
Figure 5.3 Voting ...................................................................................................... 162
Figure 5.4 Motion region segmentation ................................................................. 169
Figure 5.5 (a) Interframe difference data. (b) Enhanced current object ................. 170
Figure 5.6 (a) Motion region in previous frame. (d) Motion region in current frame. (e) Object region in previous frame. (f) Object region in current frame .............................................................................................................. 170
Figure 5.7 Object location in the edge map ............................................................ 171
Figure 5.8 Refine object contour ............................................................................. 171
Figure 5.9 Object captured from 2\textsuperscript{nd}, 3\textsuperscript{rd}, 4\textsuperscript{th}, 5\textsuperscript{th}, 6\textsuperscript{th}, 18\textsuperscript{th}, 23\textsuperscript{rd}, 35\textsuperscript{th}, 131\textsuperscript{st} frame respectively .............................................................................................................. 172
Figure 5.10 (a) $\Gamma_d$ projected on the current image. (b) $\Gamma_d$ projected on an edge map. (c) $\Gamma_e$ by background noise removal from (b). (d) The closest points in edge map corresponding to $\Gamma_d$. (e) Synthesized contour voted by $\Gamma_d$ and $\Gamma_e$. (f) The smoothed resulting contour .............................................................................................................. 174
Figure 5.11 (a) $\Gamma_d$ projected on edge map. (b) $\Gamma_d$ and $\Gamma_a$ on edge map. (c) Synthesized contour voted by $\Gamma_d$ and $\Gamma_a$. (d) Synthesized contour voted by $\Gamma_d$, $\Gamma_a$ and $\Gamma_e$. (e) Optimal contour on current image .............................................................................................................. 174
Figure 6.1 Signal decomposition ............................................................................. 179
Figure 6.2 Signal reconstruction ............................................................................ 180
Figure 6.3 Subbands of an image ........................................................................... 185
Figure 6.4 Approximate curve ............................................................................... 191
Figure 6.5 Deviation from monotonicity .................................................................. 192
Figure 6.6 (a) Subband from filter Daul4. (b) Subband from filter mna3............. 192
Figure 6.7 Watermarking and detection scheme .................................................. 194
Figure 6.8 4 walks ............................................................................................... 195
Figure 6.9 (a) Original band. (b) Modified band with sequence order. ............. 198
Figure 6.10 (a) Sort coefficients. (b) A local zoom out of (a). (c) Approximation curves. ........................................................................................................ 199
Figure 6.11 Pixel value calculation after a backward rotation ......................... 208
Figure 6.12 (a) Rotated watermarked image. (b) Reproduced image. ............... 212
Figure 6.13. (a) Reproduced image contains watermark. (b) Reproduced image contains no watermark. (c) Reproduced image contains watermark, using original image ................................................................. 214
Figure 6.14 Sub-images ....................................................................................... 216
1.

INTRODUCTION

Video is a rapidly developing and widely used multimedia due to its availability at low cost and its relevant advanced processing techniques. The expectation of automation and intelligent control in many real-life applications makes the use of video an attractive and sometimes indispensable tool. This thesis investigates mainly the object tracking in video sequences. In this chapter, a thorough literature review will be conducted, the technical challenges will be presented, and the research contributions will also be outlined at the end.

The aim of object detection and tracking is to capture an object of interest in a video sequence and keep track of this object throughout the sequence over a period of time. The applications of such techniques have been found increasingly in surveillance, human-computer interfaces, medical imaging, vision-based
control, and artificial intelligence. Their importance will become even more significant in the future.

The increasing demand within such applications seeks more efficient and effective methodologies and algorithms to make the process more practical and reliable. This has led to the continuing development of the strategies and techniques for both generic and special circumstances. These techniques also need to keep abreast with the latest multimedia developments such as the newer compression standards. This research will thus concentrate on developing new methodologies to effectively track an object in a video sequence. In addition, this work is also linked with the investigation of the copyright protection for images or videos through proposing a new and more robust watermarking scheme.

The ownership protection of the multimedia is of high demand nowadays, and the distributors will typically want to embed invisible but robust watermarks into multimedia products to claim the ownership or authorship. It is also natural that the information carried by the watermark should remain intact and detectable under image distortion, normal image processing, or even malicious attacks. This chapter will first extensively review the research development in the relevant fields.

1.1. Object detection and tracking

Object tracking in video sequences starts with the object detection that detects or recognizes the object of interest. Object detection involves finding and
isolating one or more objects from a given image. Object recognition then further identifies the object based on prior known models. The tracking is to determine the trajectory of the same object through the subsequent frames so that one can better understand the behaviour of the object within a given environment.

1.1.1 Object detection

For a video of static background, background subtraction is a common technique used for object detection. There are different ways to model the background, and a straightforward one is the averaging method in which the background is extracted by averaging the frames within a certain period (Gonzalez and Woods 2002). The moving object diminishes from the scene this way and the smooth background is derived.

The adaptive background subtraction method, alternatively, models each pixel as a mixture of Gaussians (Lee 2005; Stauffer and Grimson 2000). It can handle the repeated local motion of the object or lighting difficulties such as shading and night vision. The kernel density model (Elgammal et al 2002), on the other hand, constructs a statistical representation of the background. Heikkinä and Pietikäinen (2006) established the background statistics with the discriminative texture features. Others (Paragios and Deriche 2000) also make use of motion information, such as the inter-frame difference, to detect an object.

The detection of an object is more difficult in a video with a moving background. Object modeling is a practical method for detecting the object in the moving background. The object can be extracted by directly matching the model
with the current frame if the object model has been well established and known a priori. It however can be inefficient in situations where the template becomes complicated and varies with time due to such variables as different viewing angles and illumination conditions to the object.

An object can be described by its color, texture, shape, and by the relevant motion. The techniques for object extraction in still images, using low level features like color, texture and shape by segmentation, can also be applied to object detection in videos. In addition, the motion information is important in representing a moving object. The object detection can be feature based, or motion based, or a combination of both.

**Feature-based detection**

Feature-based object detection naturally relies on the feature extraction. The primitive features on an image are typically first extracted based on the object characteristics such as color, edge or boundary. These low level features are then carefully combined and grouped into conceptually more meaningful object features. The object is then verified or detected by comparing its features with those pre-built object features in the database. The registry of significant points and the standardization of the object (Guo) are generally required.

**Shape-based detection**

Shape-based object detection makes use of the shape properties of an object such as its boundary. The shape information should be available beforehand, and the process involves the segmentation of an object in the image as well as the
choice of the object representation. In work conducted by Jain et al (1996), the
prior knowledge of an object shape was described by a prototype template,
consisting of the representative contour or edges and a set of probabilistic
deformation information on the template. A Bayesian scheme is employed to find
a match between the deformed template and the objects in the image, and its
computational efficiency is achieved via a coarse-to-fine implementation of the
matching algorithm.

The snake model can also be employed for the shape-based object detection
and for tracking. It approximates the actual object boundary by measuring the
force due to the internal and external energy, and then tries to match with the
template. The snake model was first proposed by Kass et al (1987), and is later
extended to a few adaptive methods. Wu et al (1999) proposed an active contour
model by considering the geometric structure with an attribute vector employing
an adaptive-focus statistical model and a contour deformation mechanism. A
deformable shape template is further studied by Sclaroff and Liu (2001) to
partition the image into a globally consistent interpretation.

**Color-based detection**

A color-based approach is relatively less computational intensive but can be
easily affected by the illumination change as well as by the complexity of the
color component of the object itself. In this regard, Wu et al (1999) applied fuzzy
theory to the face detection in color images. They made two fuzzy models to
extract the skin color regions and the hair color regions, and compared them
against pre-built head-shape models to detect faces through pattern-matching method based on the fuzzy theory.

For segmenting an object of interest from a noisy and cluttered background, thresholding is most commonly used. It assumes that the image has a uniform and stationary, or at least quasi-stationary, distribution of intensities over the target and background. Such methods are however not very effective when the images contain complex and non-stationary distribution of intensities. Zhang and Desai (2001) thus utilized the multi-scale image representation of wavelet to isolate the different intensity information into different scales for such as the smooth and detail components. A Bayesian classifier and a probability density function were then used to conduct the analysis. In a similar manner, Weber and Casasent (2001) detected objects using threshold with a set of Gabor basis filters, whose parameters are separately optimized and the fusion parameters there combining the Gabor filter outputs are optimized using an extended piecewise quadratic neural network. The proper choice for the threshold is important. It is relatively simple, but easy to be affected by the noise.

Once an object template is available, locating the object can be calculated by trying to match the template at different positions on the image. The computation is usually high since it often has to search the whole image for the best fit, even though some algorithms can help reduce the search area. Moreover, the fixed-template approaches can be applied only when the object shape does not change with the viewing angle of the camera. Although the correlation method utilizing normalized cross-correlation can limit the effect due to noises and illumination
changes, it suffers from the high computational complexity. Deformable template approaches, on the other hand, seem more suitable for object detection and tracking since the object may vary with time due to its motion. In (Jain et al 1996) the probabilistic deformation information is collected on the template to reflect the object deformation. Jolly et al (1996) also defined a polygonal template to characterize a general model of a vehicle and derived a prior probability density function to constrain the template to be deformed within a set of allowed shapes. They proposed a likelihood probability density function, combining the motion information and the edge directionality, to ensure that the deformable template is restricted within the moving area in the image and the boundary of the template coincides with strong edges with the same orientation in the image.

**Motion-based detection**

The motion-based approach detects the object via the motion information (Chang et al 1997; Kim et al 1999; Wang 1998; Yau et al 2007; Zheng and Blostein 1995). Object detection is conducted by static segmentation followed by motion-based merging. In other words, the spatiotemporal segmentation, or spatial grey-level or color segmentation of a frame is first performed; the segmented regions are then grouped according to their motion characteristics. In (Kim and Chen 2003), a multiple feature space is transformed to a one-dimensional label space by using self-organising feature map neural networks, followed by an edge fusion process, in which the edge information is incorporated into the neural network output to generate the boundaries of the
segmentation. A color segmentation was further proposed in (Nguyen et al 2000), where regions undergoing similar motions are merged together. This method is known to yield more accurate results when the object is rigid and distant enough from the viewing camera. Sharma et al (2004) used wavelet directional histograms to identify simple pre-defined human actions of subjects from video frames.

1.1.2 Object tracking

After detecting the object, tracking is to monitor the object’s spatial and temporal changes in the subsequent video frames, possibly including its presence, position, size, and shape. The object in a video may vary in some way while moving in a static or moving background. It may come in different poses and shapes due to different viewing points or non-consistent illumination in the video sequence. In addition, other factors such as background complexity and object representation may also impact on the efficiency of the tracking algorithm. It is difficult for one method to deal with every situation well in the detection and tracking process. Moreover, the object in a scene often becomes inevitably occluded or hidden by the others for a certain period of time. An automatic system thus is able to recognize and track the object again. Many algorithms have been proposed in this connection, and they usually solve problems to a certain extent under specific application scenarios such as in a static background. The main tracking approaches include the use of templates (Zhong et al 2000), and feature points (Veenman et al 2001). They can be color-based or shape-based tracking, and can be integrated with the object motion.
1.1.2.1 Template match

Among all the tracking techniques, a very early and straightforward one is the template match, which dates back to the Lucas-Kanade algorithm (Matthews 2004) in 1981. The underlying assumption behind the template tracking is that the appearance of the object remains the same throughout the entire video. The template of the object is being matched at different positions in the current frame, and the position of the object is located as the one that leads to the minimum difference between the template and candidate region in the current frame (Wang et al 2002). A proper template needs to be established prior to the tracking. For instance, in (Wang 1998), the logo template is detected and made significant for the later matching steps. The template approach in fact takes both the pixel’s color and its exact position within the object into account during the matching process, unlike the histogram threshold approach, where only the colors but not the pixels’ relevant positions are considered. The drawback of the template match is that a pixel’s colors are hard wired to its relevant position, making it less flexible to handle the deformable object where the object’s pixel colors and their relevant physical positions are not fixed as in the template. Consequently some chose to update the template to more accurately reflect the most recent object (Matthews et al 2004), while others (Jain et al 1996; Zhong et al 2000) proposed the deformable template model for the problem. The motion information is then introduced into the template match in (Hager and Belhumeur 1998; Jurie and Dhome 2002). Another shortcoming for the template match is its difficulty at handling partial object occlusion as it is restricted to considering the full object
pixels as a whole. A Kalman filter is then used (Nguyen et al. 2001; Nguyen and Smenlders 2004) to make the template adaptive to the changes in object orientation or illuminations, making the template more robust against partial occlusions. Template match is generally of high computational cost since the object position of minimum matching error has to be searched for each frame for the full sized template.

1.1.2.2 Shape based approach

Object shape is an important feature that can facilitate tracking objects in a video sequence. Different representations of the object shape have been studied in the literature. The subspace model (Zhou et al. 2005), to start with, is built to capture the major shape variations in terms of a specific subspace distribution, with a Gaussian, or eigen spaces model for instance (Black and Jepson 1996). Zhou et al. (2005) constructed the subspace probabilistic models though Principal Component Analysis (PCA), and in (Moon et al. 2002) the tracking is realized through the edge-based shape detection. Some tracking methods may constrain the shape of the region to a parameterized family of shapes. The model shape can also be recovered incrementally (Sminchisescu et al. 2005) upon the newer features in its state estimation. Active contour modeling based on the assumption of strong intensity boundary of the object, in contrast, describes the object boundary through a few control points, and an early such model called the snake model was proposed by Kass et al. (1987) in 1987. Much research work has since been carried out to improve the performance of active contour methodologies.
**Active contour models**

Active contour modeling represents the object by its contour and the associated control points. The contour moves to the object boundary under the influence of image-intensity ‘force’, subject to certain internal deformation constraints. In other words, it is obtained by minimizing an energy function of color gradient, contour smoothness or other properties. Active contour modeling is suitable for describing non-rigid objects. An active contour can be represented explicitly or implicitly. In the explicit representation, the contour is a parametric curve $C(p)=(x(p), y(p))$, i.e. a function from a scalar interval $[a, b] \subseteq IR$ into the image domain $\Omega \subseteq IR^2$ (Dumitras and Venetsanopoulos 2001; Leymarie and Levine 1993; Melnerney and Terzopoulos 1995). In the implicit representation (Freedman and Zhang 2004), the contour is obtained as the zero level set $\Phi^1(0)=\{(x, y)| \Phi(x, y)=0\}$ for a scalar function $\Phi(x, y)$ defined on the image (Akgul and Kambhamettu 2003). The level set approach makes it possible to represent multiple objects as the contour splits or merges.

Brigger *et al* (2000) proposed a B-spline representation for the active contours, while others (Bertalmio *et al* 2000) used coupled partial differential equations. In (Mansouri *et al* 2004), the contour evolves through an element of a desired Lie group of plane transformations. It is linked to the contour in the previous frame through geometric transformations like translation and rotation, or an affine transformation. The watershed segmentation in (Nguyen *et al* 2002) is formulated as a minimization using topographical distance to extract the object contour as a combination of the current edge map and the contour predicted from the previous
frame. Shen and Davatzikos (2000) introduced a geometric structure via an attribute vector and integrated it into the energy function. In (Akgul and Kambhamettu 2003), the authors proposed a coarse-to-fine deformable contour optimization framework which used the scale-space and the information theory to produce a coarse representation and combined dynamic programming with the gradient descent method to minimize the contour energy. The corner detection is also used along with the contour approach in a logic programming environment (Bell and Pau 1990). In (Freedman 2003), the contour curves are constructed within a certain shape space known to describe properly the class of the objects to be tracked, and a tree-search algorithm is used in the matching process to explore the entire shape space with very little effort. The contour model in (Lai and Chin 1995) is similarly based on a stable and regenerative shape matrix which is invariant and unique under the rigid motion.

**Geodesic active contour models**

Geodesic active contour modeling was introduced as a geometric alternative for the snake model. It is a geometric model and uses energy functional minimization. The geometric active contour model is parameterization independent and allows for changes in topology easily. The main drawback is its non-linearity that results in inefficient implementations. Goldenbery et al (2001) proposed a method based on the Weichert-Romeney-Viergever AOS scheme, with the narrow band of the level set and Sethian’s fast marching method, to implement a fast version of the geodesic active contour model. Paragios and Deriche (2000) on the other hand, proposed to employ the motion detection and
geodesic active contour to track multiple moving objects. They approximated the probability density of the background and movement by a mixture model through the inter-frame difference, and integrated them into an active contour objective function which is then minimized using a gradient descent method. The Graph partitioning method for the global minimization can also be found in the curve evolution (Boykov and Kolmogorov 2003; Sumengen and Manjunath 2006; Xu et al 2003).

Although most contour approaches are based on significant surrounding edges, some can be based on regional features (Chan and Vese 2007; Mukherjee et al 2004; Sumengen and Manjunath 2006). In (Mukherjee et al 2004), the region homogeneity within the boundary was used in the energy functional for the leukocyte detection. The method automatically identified and tracked the cell by exploiting the shape and intensity characteristics of the leukocyte cells.

**Framework for filtering**

The tracking problem in many applications can be formulated as filtering in the state space. Given a state sequence \( \{x_k\}_{k=0,1...} \), and the observed measurement \( \{z_k\}_{k=1,...} \), the state \( x_k \) or probability density function (pdf) \( p(x_k|z_{1:k}) \) is estimated from \( z_{1:k} \). The Bayesian inference provides the solution via calculating the posterior pdf \( p(x_k|z_{1:k}) \) with the likelihood \( p(z_k|x_k) \) and the prior pdf of the current state \( p(x_k|z_{1:k-1}) \) often derivable from the known \( p(x_{k-1}|z_{1:k-1}) \) and the dynamic equation of the form \( x_k = f_k(x_{k-1}, v_k) \). A number of filtering algorithms based on the filtering framework have been reported in the literature. The Kalman filter as a recursive linear estimator was first integrated into the snake model in (Baumberg
and Hogg 1994; Boykov and Huttenlocher 2000; Peterfreund 1999). The Extended kalman filter (Rosales and Sclaroff 1999) was largely used to estimate a 3D object trajectory from the 2D image motion. Particle filtering, another technique based on the Monte Carlo integration, was introduced by Isard and Blake (1998). They proposed a Condensation algorithm based on factored sampling in the B-spline representation of the curve, and modelled the curve with the prior probability density, together with the observation that characterises the statistical variability in the current image, to estimate a posterior distribution. It is more effective than Kalman filters for a cluttered background. The Condensation filter is also used in (Verma et al 2003) for tracking multiple faces. The particle filtering algorithm has also been employed (Rathi et al 2007) to realize the geometric active contour via the level set models to track the deformable object, incorporating the importance sampling (IS) density in the contour model. Later in (han et al 2005), the importance sampling in the particle filter was replaced by the Markov chain Monte Carlo (MCMC), which included a Markov random field (MRF) motion prior to the multi-target tracking. Li and Chellappa (2000) also proposed a simultaneous tracking and verification method based on the particle filters that are applied to vehicles and faces, while Rui and Chen (2001) proposed to track the face contour based on the unscented particle filter. The Hidden Markov Model (HMM) was adopted by Chen et al (2001) for tracking through the JPDAF data association, and the Probabilistic Data Association Filter (PDAF) was also employed in (Hue et al 2002; Rasmussen and Hager 2001). Cham and Rehg (1999) proposed a probabilistic multiple-
hypothesis framework for tracking complex targets with high dimensional state spaces. It is a classical approach to represent multimodal distributions with Kalman filters. They explicitly modeled the modes to avoid the large number of samples that a Monte-Carlo-based scheme requires, and used a sampling-based state space search process to generate a set of hypotheses corresponding to the local maxima in the likelihood to avoid the need for a complex figure detector required by classical MHT methods. However, the Monte Carlo approach has two major limitations. Firstly, the accuracy improves linearly with the number of samples and, secondly, more samples are required if the integrand has peaks in some small regions while remaining very small elsewhere (Li and Chellappa 2002). The sequential importance sampling (SIS) was subsequently adopted there to improve the performance. In the SIS methods, the dynamic density is in fact approximated by a set of its properly weighted samples. In (Mansouri 2002), the tracking of the object region was formulated as a Bayesian estimation problem and the tracking algorithm was determined through a partial differential equation for the level set. The formulation was further extended to allow the inclusion of additional information such as the priors on the region’s intensity boundaries. It avoided the motion field or the relevant motion parameters, required very little a priori information about the region, and made no assumption on the strength of the intensity edges of the region.

**Partial occlusion**

Contour based tracking uses the visible boundary of the object. It may come from a visible part of a partially occluded object. The partial occlusion is a
difficult problem in tracking and has received some attention in (Darrell and Covell 2001; Fu et al 2000; Ricquebourg and Bouthezmy 2000; Yilmaz et al 2004). Fu et al (2000) employed a piecewise contour prediction, using the local motion and color information on both sides of the contour segment, to handle the occlusion boundary. In (Darrell and Covell 2001), a cumulative similarity measurement which characterized the shape of the local image homogeneity was used to enable tracking near the occluding boundaries. Gentile et al (2004) further proposed a segmentation method based on the correlation between the features of certain parts to make optimal use of the good features for the tracking robustness to the occlusion. Wu and Yu (2006) used a Boltzmann distribution for the prior shape variation in the two-layer statistical field model and embedded this distribution and the image likelihood into the Markov field. This has made the method more robust to partial occlusions and clutters.

For contour based tracking, we note that the traditional methodology suffers from the slow convergence because of a large number of control points involved, as well as the difficulties in determining the weight factors associated with the internal energies of the boundary.

1.1.2.3 Color based approach

The color feature is another main aspect to enable accurate tracking without the explicit use of edge related features. The statistical distribution of the color can effectively characterize the object for the tracking purpose. The probability density functions are often established through proper models to describe the observed object. The classical parameterized models such as the Gaussian model
(Darrell et al 1998) typically have low space complexity and require relatively small training sets, in contrast to the histogram (Brichfield 1998) approach that would need much more training data. When multimodal densities are involved, in a Gaussian mixture model (Wu and Huang 1999) for instance, the type and the number of mixed functions as well as the model parameters need to be properly solved. The Expectation-Maximization (EM) algorithm is well known for solving the mixture density (Sigal et al 2004). On the other hand, the non-parametric approach for the density estimation is also attractive since it can be applied to arbitrary distributions and does not have to assume the form of the underlying densities. The kernel density estimation (KDE) is one of such non-parametric techniques.

For the non-rigid objects, Comaniciu and Ramesh (2000) adopted the mean shift method to track them, while Bradski (1998) presented a continuously adaptive mean shift CAMSHIFT system in a perceptual user interface to track faces. Xu et al (2003) proposed an adaptive approach using fast color thresholding and region merging while improving mean-shift’s robustness. Comaniciu et al (2003) proposed a kernel-based object tracking, where the target representations based on feature histogram are regularized by the spatial masking with an isotropic kernel. The mean shift procedure is then applied to find the local maxima. Fan et al (2007) recently proposed a kernel-based tracker based on the motion field representation. A joint feature-space model is also possible (Elgammal et al 2003; Yang et al 2004), and the trust-region methods (Liu and Chen 2004) can be used for optimization in the tracking. They modelled the
object representation with both color and edge information via two coupled weighting schemes derived from a covariant ellipse model. The kernel density estimation is a powerful technique but is in general computationally intensive. In order to reduce the computational cost associated with the kernel approach, Elgammal et al (2003) proposed a fast Gauss transformation for the kernel density estimation. Likewise in (Gurwicz and Lerner 2005), Gurwicz and Lerner used a spline to smooth the density so that only a few coefficients rather than the whole training set are required to be estimated, resulting in the reduced computational cost in the implementation of the Bayesian network classifier with KDE.

To model the appearance change of an object, Jepson et al (2003) proposed an adaptive appearance model for tracking complex natural objects. The model consists of three components; a stable component that is learned from a relatively long duration of frames, a two-frame transient component that captures the latest changes, and an outlier process as the third component. The parameters of the mixture model are learned there with the EM algorithm. This model combines these components with proper weighting to achieve a better tracking result by relying on more stable components. Elsewhere in (Hager and Belhumeur 1996), the appearance of an object was explicitly modelled with its geometry and illumination changes, and in (Fablet and Black 2002), a view-based representation (instead of object appearance or static background) by 2D optical flow was created to track human motion. Collins et al (2005) however approached the problem in different way. They presented a mechanism of online
feature selection to evaluate multiple features for the tracking, using the log of the likelihood ratio of the conditional sample densities from the object and from the background. The feature evaluation mechanism is in fact embedded in a mean-shift tracking system. It is known that an object appearance may also be affected by its shadows, and some of the work has been done (Kato et al 2002; Matsushita et al 2004; Wang et al 2006) to explicitly address the shadow effects. The HMM in (Kato et al 2002), for instance, was used in the segmentation there to remove the shadows from the cars.

In the pursuit of a different perspective, Avidan (2004) integrated the support vector machine (SVM) classifier into an optic-flow-based tracker. The tracker can deal with the partial occlusion better because the new frame is matched against all the patterns the classifier was trained on rather than merely the previous frame. Williams et al (2005) also proposed to use the relevance vector machine (RVM) to locate an object. The RVM, a sparse learning algorithm, is trained to learn the relationship between the local motion of an object and its appearance in an image. The sparsity of the RVM makes the estimation more efficiently and can thus be used in real time videos.

1.1.2.4 Motion integration

Motion has been integrated in object tracking in a variety of work, see for example (Fu et al 2000; Vazquez et al 2006; Zhou et al 2005). Motion is often modeled as either an affine motion (Weiss and Adelson 1996) or a 2D projective motion (Torr et al 1999). A dynamic layer representation of scene was introduced by Tao et al (2002) to track objects within the frame of Bayesian estimation. The
motion layer was modeled as a 2D rigid motion with only 2D translation and rotation components. In (Jain and Jain 1981), the minimum description length principle (MDL) was used to automatically select the best number of motion layers to facilitate the edge tracking between the frames. In (Paragios and Derich 2000), the motion estimation from the inter-frame difference was integrated into the contour approach to capture the moving object. The Markov random fields (MRF) model was also considered in (Paragios and Tziritas 1999), and in (Grinias and Tziritas 2001), a motion-based segmentation via region growing algorithm was presented for the tracking purpose. On a different footing, Wechsler et al (2004) applied the Statistical Learning Theory (SLT) to the motion estimation in this regard, because the SLT provided the analytic generalization bounds for the statistical model selection. Alternatively Mujica et al (2000) provided a motion parameter estimation (ME) algorithm based on the spatio-temporal continuous wavelet transform (CWT). This CWT-based algorithm sequentially optimizes the motion parameters by maximizing the associated energy densities on the frame-by-frame basis, thus allowing it to track a moving object. Veenman et al (2001) on the other hand introduced individual, combined, and global motion models to fit the existing qualitative solution in their framework for the motion correspondence problem, which is integrated with the greedy matching algorithm.

1.2. Watermarking

Watermarking is to embed certain data, such as authorship or other information, into a digital image or video, and the watermark should be
detectable later on for its existence. With the development of digital technology and the Internet, multimedia distribution has become more and more wider spread. The need for the copyright protection of the multimedia property is thus increasingly prominent, and watermarking is one of the main solutions for this purpose.

The watermark on an image, and directly or indirectly on a video, can be visible or invisible. The main requirements for an invisible watermark are typically its imperceptivity, robustness, and security. The imperceptivity implies that the embedding of the watermark will not cause any visible degradation of its hosting image. The robustness indicates that the watermark can survive normal image processing and resist certain degree of attacks including such as image geometric distortion and compression, image noise injection, and other malicious attacks. The security refers to the fact that a non-authorized user can not create, remove or modify the watermark easily if at all (Katzenbeisser et al 2000).

The design of a watermarking system usually needs to take into account a number of aspects in order to maximally satisfy the requirements of a sound watermarking system. They include the watermark construction, the amount and the location of the watermark, the embedding process, the security consideration, and the extraction of watermark for the detection. The amount of watermark is often related to the invisibility of the watermark, while the location of watermark would impact on its invisibility, robustness and security. A sound watermark is thus expected to resist distortions caused by such as lossy compression, white noises, low-pass filtering, watermark energy change and geometric distortions.
1.2.1 Spatial domain

The watermarking can be performed in the spatial domain or in a transform domain. The watermark in the form of an \(m\)-sequence-derived PN code, for instance, can be embedded (Van Schyndel et al 1994) in the least significant bit (LSB) plane of the image data. The two methods by Bender et al (1995) are very illustrative of watermarking in the spatial domain. In the first method, the Patchwork method, a number of pairs of pixels \((a_i, b_i)\) are randomly selected to embed a bit 1 by increasing the \(a_i\)'s by one and decreasing the \(b_i\)'s by one. The expected value of the sum of the differences between the \(a_i\)'s and \(b_i\)'s of \(N\) pixel pairs is given by \(2N\). The second method, the Texture Block Coding, embeds the watermark by copying one image texture block to another area in the image with a similar texture. Then the watermark is detected by autocorrelation function. This method is reasonably robust to distortion since both image areas are distorted in a similar way. The disadvantage is that the image must have such texture areas. Similar to the patchwork algorithm, Pitas (1996) and Pitas et al (1995) proposed to cast a signature onto the digital image. The watermark there consists of a binary pattern of the same size as the original image, where the number of “ones” is equal to the number of “zeros”. The image pixels are divided into two sets of equal size according to the watermark. Then the watermark is superimposed by changing the number of elements in the set of ones, and the watermark detection is based on a statistical test. Langelaar et al (1997) also proposed a block base spatial watermarking method by extending the above idea, and the performance is further improved by the pixel classification proposed by
Bruyndonckx et al (1995). Voyatzis and Pitas (1996), on the other hand, watermarked their images by inserting logo like patterns with torus automorphisms. Moreover, watermark embedding based on quantized index modulation (QIM) rather than the spread-spectrum modulation has also been well considered in (Chen and Wornell 1999).

1.2.2 Transform domain

The watermarking on transform domain has been demonstrated to be more robust against most of attacks (Cox et al 1997) than on the spatial domain. The associated transforms can be the discrete cosine transform (DCT), discrete Fourier transform (DFT), discrete wavelet domain (DWT), and so on. Ruanaidh et al (1996) proposed a watermarking scheme through phase modification in the frequency domain. In (Ruanaidh and Pun 1997), a watermarking technique invariant to translation, rotation, and scaling has been studied. It is a hybrid between the DFT and the log-polar mapping.

Discrete cosine transform

Koch and Zhao (1995) considered the watermarking on the DCT domain. An image is divided into blocks of size 8x8, and pairs of mid-frequency DCT coefficients are selected from pseudoly random blocks. These coefficients are then modified accordingly to embed the watermark bits. Swanson et al (1996) also proposed a DCT based watermarking method using the frequency masking on the DCT blocks, while Tao and Dickinson (1996) proposed an adaptive watermarking technique on the DCT domain based on a regional perceptual
classifier with the assigned sensitivity indexes. Podilchuk and Zeng (1997)
however used only the noticeable difference to determine an image-dependent
watermark modulation mask for perceptual watermarking.

Langelaar and Lagendijk (2001) proposed the differential energy watermark
(DEW) algorithm that embeds the watermark bits by selectively discarding high
frequency DCT coefficients in the compressed data stream. The performance
there can be controlled by the number of DCT blocks that are used to embed a
single watermark bit, the maximal coarseness of the quantizer used in pre-
encoding that controls the robustness of the watermark against re-encoding
attacks, and the lowest DCT coefficients that are allowed to be discarded. Cox et
al (1997) proposed to embed the watermark in the DCT domain through spread
spectrum communication, and in fact they embedded the watermark in the low-
frequency components of the image for better robustness. Huang and Wu (2000)
investigated the use of the Genetic algorithms for the watermarking in the DCT.
The watermark in the form of a recognizable pattern is embedded into the
selected middle frequencies of the image, and the genetic algorithm is applied to
choose the DCT blocks for the embedding to optimize the quality of the
watermarked image. The watermark capacity has been addressed in (Barni et al
2000; Moulin and Mihcak 2002), where Barni et al proposed to evaluate the
watermark capacity for the full frame DCT image in a non-additive, non-
Gaussian framework, and Moulin et al introduced a framework to evaluate the
watermark capacity using an information-theoretic model for the image.
Discrète wavelet transform

The DWT is known to provide multi-resolution for the image watermarking (Barni et al 1999; Kundur and Hatzinakos 1997). Dugad et al (1998) pioneered the spread spectrum method for the image watermarking in the DWT domain by embedding the watermark into high frequency subbands with a constant weighting factor to achieve the invisibility, but it is not robust to common image processing. Kutter and Winkler (2002) hence proposed a perceptual model that takes into account the contrast sensitivity and texture masking to improve the performance of the spread-spectrum watermarking. In (Hsu and Wu 1998), both the host image and the watermark are decomposed to multi-resolution representations, and the watermarks of different resolution are embedded into the corresponding resolution of the decomposed images. Kundur and Hatzinakos (1998) embedded the watermark in binary code by suitably quantizing some of the coefficients of the detail bands, while Zhu et al (1999) modified the DWT coefficients of the high-pass bands proportionally to their magnitude. Lu and Liao (2001) proposed to quantize a host image’s wavelet coefficients as the masking threshold units, and then embed two complementary watermarks using cocktail watermarking. The extraction of the watermark does not need to access the original host image in this case.

Kwon and Tewfik (2002) proposed an adaptive watermarking scheme in the multi-wavelet transform domain using successive subband quantization and a perceptual model. However, Ghouri et al (2006) investigated a more balanced multi-wavelet transform. An adaptive perceptual model based on image
properties is considered to fulfill the imperceptivity requirement and the principle of spread-spectrum communications is adopted to achieve the watermark robustness. Kumsawat et al (2005) also proposed a semi-blind watermarking scheme in terms of the discrete multi-wavelet transform and used generic algorithm optimization to improve the performance. A blind image watermarking algorithm based on the multi-band wavelet transform and the empirical model decomposition can be found in (Bi et al 2007). The watermark is embedded into the mean trend of each sub-image in the multi-wavelet domain for better performance since the mean trend is quite stable under the attacks of high frequency noises. Bao and Ma (2005), on the other hand, presented a watermarking scheme using a quantization-index-modulation and the singular value decomposition in the wavelet domain, although the method is sensitive to filtering and random noises. In the pursuit of a different perspective, Ng and Gary (2005) proposed a scheme of maximum-likelihood detection in which the distribution of the image DWT coefficients is modelled by a Laplacian probability distribution.

**Human visual system**

Based on the sensitivity differences of the human visual system (HVS) for the different frequencies, e.g. high frequency is often less noticeable than low frequency to human eyes, HSV has been used to achieve robust and invisible watermarking (Piva et al 1997; Tao and Dickinson 1996). The amount of the watermark to be concealed can be designed to reach the maximum within the invisibility constraint. Barni et al (2001) proposed to watermark in the DWT
domain by exploiting the characteristics of the HVS. The watermark is masked pixel by pixel by taking into account the texture and the luminance content of the image subbands. In (Wang et al 2002), the middle frequency bands were used to achieve both perceptual invisibility and robustness to compression since lossy compression schemes often eliminate high frequency components. Chou and Liu (2003) proposed a watermarking scheme for color images based on a color visual model which is capable of estimating the just-noticeable distortion (JND) profile of each wavelet subband in the $YCbCr$ color space. The blocks of wavelet coefficients having higher JND energies are chosen for watermarking and the embedding process is carried out by quantization index modulation of the wavelet coefficients.

**Geometric distortion**

Some watermarking algorithms explicitly addressed the capability of the watermark to resist geometric attacks. Alturki and Mersereau (2000), for instance, used an iterative search technique to cope with the geometric attacks. Sebe and Domingo-Ferrer (2001) presented an oblivious watermarking scheme using a tile-based embedding technique in the spatial domain, allowing the watermark recovery from a scaled or geometrically distorted watermarked image. Pereira and Pun (2000) chose to embed an additional template besides the watermark into the DFT domain of the image, as the local peaks in predefined positions. This embedded template was used to estimate the affine geometric attacks in the image. This type of approach can however be tempered by certain malicious attacks (VoloShynorskiy et al 2001). To overcome this problem, a
periodic pattern was inserted into the spatial domain in (Deguillaume et al 2002; Voloshynovskiy et al 2001), although it is still easy to detect and remove the watermark by an attacker. Delannay and Macq (2002) thus engineered an informed coding approach to further improve the security. They modelled the resynchronization pattern with an image-dependent secret binary mask to prevent the exposure of the specific peaks in DFT. In (Bas et al 2002), on the other hand, feature points were made use of to resist geometric attacks. The salient feature points firstly extracted from the image were used to define a number of triangular regions, and the watermark is then embedded into each triangle by an additive spread spectrum scheme. However this method requires the robust detection of the feature points in the image in order to retrieve the watermark, and the feature points are not protected by a secret key, making it vulnerable to malicious attacks. Dong et al (2002) resorted to the original image to compensate for the geometrical deformations induced by Stirmark. A regular triangular tessellation is applied to both the original and the watermarked image. The vertices of the watermarked image are then adjusted based on the minimum error between each original triangle and the corresponding triangle in the attached image. Others also made use of the invariant transform (Lin et al 2001; Zheng et al 2003) or moment based normalization (Alghoniemy and Tewfik 2004; Dong and Galatsanos 2002) to achieve resilience against geometric distortions, and Coltuc and Bolon (1999) used the histogram specification to make the watermark invariant to geometrical distortions. Lin et al (2001), with a different perspective, suggested embedding the watermark in a one-dimensional signal obtained by projecting the Fourier-Mellin
transformed image onto the log-radius axis. However their methods remain vulnerable to cropping attacks. Dong et al (2005) recently presented two watermarking methods in the DCT domain that were robust to geometric distortions. In the first method, the image is normalized to meet a set of predefined moment criteria, so that the image remains invariant to affine transform attacks. The second method is based on a watermark resynchronization scheme which can alleviate the effects of random bending attacks. A deformable mesh is later used to correct the distortion caused by the attacks. In (Seo and Yoo 2006), the invariant regions of an image were used for watermarking. The watermark is embedded after the geometric normalization. Dugelay et al (2006) proposed a blind watermarking algorithm to offset the local geometrical distortions. They inserted a pre-defined additional piece of information modelled in the same way as the information bits embedded into the image in the spatial domain. These resynchronization bits are used to estimate and compensate for the small local or global geometrical deformations. In (Tang and Hang 2003), a combination of feature extraction and image normalization was proposed to resist geometric distortion and signal processing attacks.

1.2.3 Video watermarking

Since video sequences consist of consecutive frames, intuitively, the watermarking techniques for images can also be applied to video frames. Nevertheless video watermarking has some differences to image watermarking and presents particular challenges. One of the important differences is that the amount of data in a video is much larger than in an image. The computational
cost becomes a more important factor to be considered for video watermarking. On the other hand, the invisibility requirement can be less significant in a video since the frames rate helps to reduce the visibility of watermark to human eyes. Many videos employ lossy compression techniques, which would remove spatial, temporal and perceptual redundancy from the video. As a consequence, the compression may destroy the watermark. Moreover, the watermarking design for videos has to consider additional attacks to the frames such as frame dropping and frame swapping.

The classical approach to watermark a compressed video is to decompress the video, perform watermarking in spatial domain or transform domain, then recompress the watermarked video. Swanson et al (1997) proposed a multi-scale watermarking method for the uncompressed videos. A wavelet transform was applied along the temporal axis of a video to obtain a multi-resolution temporal representation of the video. The low-pass frames consist of the static components in the video scene while high-pass frames capture the motion components and changing nature of the video sequence. The watermark embedded in the low-pass frames exists throughout the entire video scene and the watermarks embedded in the high-pass frames are localized in time and change rapidly from frame to frame. Thus the watermark consists of static and dynamic components. The method is able to resist frame averaging attack and allows watermark to be detected from a single frame. Zhu et al (1999) proposed to embed the watermark, modelled as an i.i.d. Gaussian random vector, to all the high-pass bands in the wavelet domain for the image or video. The hierarchical nature of the wavelet
representation allows multi-resolution detection of the watermark. Thus the
detection of lower resolution watermarks reduces the computational complexity.
In (Lancini et al 2002), the watermarking is again conducted in the uncompressed
domain. Three different masks corresponding to the luminance, texture and
temporal characters are used to improve the watermark invisibility, and an error
correcting code method is employed for the robustness.

Watermarking in an uncompressed domain requires the watermark to be
inserted with excessive strength to resist the compression distortion. It is also
computationally expensive (Alattar et al 2003). Watermarking in a compressed
domain can make use of the compression parameters to adjust the watermark
embedding to improve the watermark robustness, invisibility and capacity. In
(Simitopoulos et al 2001), the watermark is directly embedded in the compressed
video streams of MPEG-1/2 with a perceptual model, while the watermark
detection is performed in the compressed domain without the use of the original
video. Alattar et al (2003) proposed a video watermarking method in the MPEG-
4 compressed domain. The spatial spread-spectrum watermark is embedded
directly into the compressed MPEG-4 bit-streams by modifying the DCT
coefficients accordingly. A synchronization template is created there to combat
geometric attacks, and a drift compensator is adopted to prevent the accumulation
of the watermark distortion and to reduce the self-interference watermark.
Langekaar and Lagendijk (2001) presented a compressed domain watermarking
technique to insert the watermark into the DCT coefficients according to the
differential watermark energy. Watermarking in a compressed domain is closely
related to the compression method, and the size of the watermark has to be adjusted for the invisibility without having to decompress the video first.

The problem of geometrical attacks for video frames has been studied in (Alattar et al 2003; Serdean et al 2002; Wang and Pearmain 2006) using similar methods as in the image watermarking. In particular, a synchronization template is considered in (Alattar et al 2003) while an additional spatial domain reference watermark is utilized in (Langelaar and Lagendijk 2001).

The object-based video watermarking has also been investigated in the literature. Swanson et al (1997) employed a segmentation algorithm to extract objects from the video, and then embedded a unique watermark into the segmented objects according their perceptual characteristics. Objects that are similar visually in the nearby frames may use the same watermark with small changes according to the object transformation. If the object changes too much, a new object and new watermark are then defined. All objects defined in the video are collected into an object database, and the watermark is detected via a ratio test on the generalized likelihood. This method can protect objects against statistical analysis and averaging, and may be incorporated into the MPEG-4 object-based coding framework. A simplified block-based (MPEG) approach is thus implemented in the DCT domain for the object watermarking. Barni et al (Barni et al 2000) further proposed to embed a watermark into each object of an MPEG-4 coded video bit stream by imposing specific relationships in the predefined pairs of quantized DCT middle frequency coefficients in the luminance blocks of pseudo-randomly selected macro-blocks. The watermark
may be destroyed if the video is converted into other compression format since it is embedded into compressed a MPEG-4 bit stream. Piva et al (2001) suggested to embed the watermark into the DWT of each frame before compression. They decoded the MPEG-4 video bit stream, and extracted the objects from the decoded frames. The object images are then decomposed into the DWT domain and are embedded as the watermark into three detail subbands. On the other hand, Vassaux et al (2002) proposed a scrambling technique that adapts the classical watermarking scheme of spread spectrum to operate in the spatial domain of the MPEG-4 video. They used the scrambling to spread video object pixels obtained from MPEG-4 on the full image of the frames.

1.3. Challenges

Tracking an object by feature points typically requires that the object’s shape remains consistent in the movement of the object and the necessary feature points are continually available and distinctive. Shape modelling for the object and the registration of the feature points are also required. Although the method of tracking by feature points is relatively fast since it largely deals with the matching of the corresponding feature points, it may run into difficulties when handling the problems of partial object occlusion and shape inconsistency, limiting its potentially for wider applications in the real world. The active contour method, on the other hand, is based on the assumption of strong intensity of the object boundaries against the background, which is true most of the time when we observe the world, except when under weak illumination such as in a dark scene, or when the object happens to be in the same or very similar colors as
the background. An animal camouflaging itself for self protection is a typical example of the latter. The active contour method tracks the object of interest by its boundaries rather than the region itself within the boundary, and it doesn’t concern itself with how the object looks as long as its boundaries are distinctive and detectable, even if only a part of the object can be observed at times. Although an efficient implementation of this method in the recent time has used the level set to represent the multiple objects at any time and used a fast marching algorithm to reduce the computational cost, when propagating the initial contour to the object’s boundary, it is difficult to identify the objects and track them when the object’s boundary does not distinguish much from a cluttered background. Also it is difficult to identify the shade from the object in contrast to the case for the human eyes, largely because the human has the knowledge and experience to interpret the vision. The properties of the object region can also intuitively help in identifying the shade of the object, because they will characterise the object with richer information. The problem of shade or partial occlusion may be solved by investigating the available consistent sub regions. However tracking by the whole object region is typically computational heavy since often all the pixels within the object will participate in the calculation of the motion parameters or the pixels’ optical flows. Some algorithms for handling partial occlusion have consequently become very complicated when involving all the elements in the region.

There are a few considerations for an object tracker of good performance. They are the speed, accuracy and generality. Many algorithms in this area focused on
one or certain particular aspects of the application to achieve an acceptable performance. In a real-time application, on the other hand, the speed is more important. Also the accuracy is always critical for tracking the object in an application, regardless of how well the description of the object’s appearance may be required. The more difficult tracking situation is when the object has a varying shape with a changing background, since the changes in the video sequence are dynamic and illumination on the scene may vary significantly. An effective modelling of the object, with distinctive and consistent features, would be of great help to improve the tracking accuracy.

1.4. Research objectives

The main objective of this research is to develop an efficient and practical mechanism for capturing an object of interest throughout a video sequence. The method will be sufficiently robust to illumination variation, shape changing, partial occlusion, shade appearing, cluttered and moving background, and will have a lower failure rate in a difficult tracking environment such as when the tracked object resembles the background in color. The accuracy on the capture of the object will be a major goal of this thesis. Since a complete solution in every aspect of the object tracking problem is way beyond the scope of a single theses, our stress will be placed on achieving better tracking accuracy for the potential application. We will investigate a scheme of dynamic analysis and optimal mechanism, utilizing existing advanced techniques while exploring the new. The computational cost will also be taken into account and made relatively low. Ultimately, we expect to develop a tracking scheme of high performance that is
to work well under various unfavourable critical conditions and with a moving background.

The second objective of this research is to devise a robust and efficient watermarking scheme that could be selectively applied to certain image regions or frames or even some specific objects in the video sequence. The watermark can not only provide evidence for copyright verification, but also supply certain amount of object information along the frames for the users. Although some attempts have been made on object protection in videos via the object accessibility provided by MPEG-4, they depend heavily on the compression standard and are difficult to mix with other strategies. Despite the fact that the core of our proposed watermarking scheme is largely considered image wise, we expect that such a watermarking system can potentially provide a certain degree of object protection within a video regardless of its compression format. Based on the work on the object tracking, the object locations could even be embedded as partial watermark and used as a cue to locate the object in the detection process, although such an application is way beyond the current research scope, the watermark will be made robust to survive certain distortions caused by such as rescaling, cropping or rotating the images or the contrast changes.

We note that all the research components each correspond to an important aspect of the target environment and the associated specific methodology so that the object tracking can be both adaptive and dynamic. To explore the advantages of a specific environment, efforts have been made to target accordingly the object appearance, discriminating colors, as well as to the more challenging object
camouflages. In fact a voting scheme is proposed to potentially combine such individual aspects together. The object watermarking, although it can be largely on its own, should be considered as the first effort towards embedding search cues in the processed videos. In other words, these research components will well blend together in an ultimate application system.

1.5. Contribution

The main contribution of this thesis lies in the proposed and development of the kernel based object appearance model, the Bayesian framework of discriminative features, and the concept of dominant feature elements, for tracking an object or a camouflaged object in a video sequence, and also lies in developing a non-traditional watermarking scheme for the video frames. Methodologies have been designed to survive the environment of non-rigid object shape, illumination variation, local appearance change and partial object occlusion, and can be made more flexible, efficient and of lower computational cost. We note that our experiments on single objects have yielded convincingly good results while multiple objects of similar shape and colors are excluded from our current scope of the research. The proposed watermarking scheme pivots on the new and non-traditional concept by this author that the watermark be embedded in the choice of the decomposing wavelet filters, with its robustness well illustrated.

Kernel density approach

More specifically, we employ the kernel density estimation to develop the
object appearance model for the tracking purpose. We statistically model the
object appearance in an effective manner, where the object colors are
incorporated with their spatial relevance directly or indirectly to represent the
object. This model capitalizes on patches of stable object appearance and
simplifies the object representation while retaining sufficient information that
characterizes the object. Since an object is associated with its color probability
density function across a set of pixel regions, we for the best directional
uniformity propose to partition the object in terms of the base shapes such as the
concentric annuli or polygons. We design the object representation to allow the
independent processing of the color features while at the same time making the
implicit use of location information without having to involve additional model
parameters. We then introduce the weighting factors to differentiate the
significance of the relative physical locations when measuring the similarity of
two probability density functions, which facilitates the tracking of the more
robust object features. We also propose an adaptive template, and abstracted the
object appearance by structured statistical samples that are generated by the
kernel function on the respective local regions. The number of such samples is
subsequently much less than that of the object pixels while still retaining
sufficient object characteristics. One advantage of this template is that it caters
for the object deformation naturely in its representation, and resists well the
potential illumination variation.

**Discriminative feature approach**

We further investigate the design and use of the discriminative object features
for efficient object tracking. We defined the discriminative features by means of the discrimination of the object in intensity density from its local background in a video of moving background. We forge a Bayesian framework in which the object pixels in the next frame can be derived directly by using the discriminative portions of the densities from the chosen color space. We in addition devised the concept of dominant feature elements and used them to capture the object through the dynamic extraction of the dominant elements for the object template, and we in fact segmented the object template via optimal intensity bands to best distinguish itself from the local background. We also applied the spatial filtering and contour adaptation to further refine the object location and shape, as is shown in the implementation where the effectiveness is also manifested in capturing the non-rigid object in a moving background.

**Tracking camouflaged object**

In pursuing the difficult task of tracking a camouflaged object within a background of very close color in a video sequence, we proposed the method of Weighted Region Consolidation to iteratively strengthen the object vision when the colors for both the object and the background are similar and unsteady with large noises. We first synchronized the noisy background to detect the object motion by a model of mixture density on the frame difference, and to model the motion pixels whose model parameters are determined through the Expectation Maximization algorithm. We then enhanced the motion pixels and the candidate object area by their distribution probabilities and the cohesion of the pixel neighbours. In fact, we consolidated the object region by evaluating for each
pixel its weighted overall neighbourhood probability based on the pixel distances and probabilities. In order to make use of the partial data on the object contour from several different sources, we also proposed a voting scheme which reconstructs a more trustworthy object contour from the coarse contours obtained by different means. More precisely, we designed the voting scheme to extract a more accurate object contour by synthesizing those derived from several approaches with different levels of local confidence. The confidence on a contour indicates the reliability of segments of the contour generated through such as edge maps, motion detection or color segmentation, and reflects how well the conditions that underpin the associated algorithms are met near the corresponding segments.

**Filter-altering watermarking**

We finally proposed a watermarking scheme for copyright protection for the video frames, and potentially for even embedding certain object searching cues in the video. We watermarked the frame images by encoding the watermark bits into the choice of the wavelet filters, in contrast to the conventional watermarking methodologies. We selected such filters from different wavelet classes in such a way that they lead to sufficiently distinguishing subbands upon different sequence of filters. We developed a partial sorting on a chosen subband to mark the watermark existence, to better camouflage the watermark, and to fend off certain noise injection attacks. The selection strategy of the subband coefficients for the partial sorting can be user defined and aims to achieve better imperception in the selected band. We then devised a detection scheme that
detects the watermark by measuring the deviation of sorted coefficients specifically scattered in a targeted subband. Experiments conducted on still images and video frames show that the proposed scheme is robust to both noises and illumination changes.

1.6. **Organization of the thesis**

This thesis is organized mainly as follows. In Chapter 2, we first investigate the kernel-based object appearance modeling for object tracking. Then in Chapter 3, we develop an additional density propagation framework to best make use of the discriminative pixel densities in the local background of the video scenes. Chapter 4 is then allocated to devising a general and practical strategy that utilizes dynamically the dominant feature elements from the frames of moving background, while Chapter 5 is dedicated to pursuing tracking camouflaged object that bears very similar colors as its background. In addition, a novel watermarking scheme based on the dynamic choice of wavelet filters is investigated in Chapter 6 for the video frame images. The final conclusion is summarized in Chapter 7. The research publications arising from the work done in this thesis are also summarized in the Publication page just ahead of the references.
2. KERNEL-BASED FEATURE MODEL

2.1. Introduction

Searching for a target region that matches the object template in the previous frame in a video is somewhat different from object recognition (Giannarou and Stathaki 2007; Taubin and Cooper 1992; Pand and He 2008; Neil and Curtis 1996; Melo et al 2006) in its traditional sense. The main task in this process is to locate the position of an object, coming from the previous frame, in the current frame. Such an object usually has little shape or color changes, this and the background information can all be utilised for the tracking purpose. Sometimes it may additionally require detailed description of the object such as the shape deformation. We will here describe our first effort, on the kernel based feature
modeling, to meet the challenges to be faced throughout this thesis towards the object tracking.

The main approaches considered so far in the literature for object tracking are region based, e.g. template matching, or the contour based, e.g. snake model. The template matching (Jain et al. 1996) is to match the whole object region directly at different positions in the current frame to locate the new object position that leads to minimum errors, while the contour approach (Goldenbery et al. 2001) emphasizes the extraction of the object contour based on certain deforming criteria such as the minimization of an energy function. Template matching, though intuitive, has limitation on processing objects of deformable shapes, even though certain deformable template methods have been proposed (Jain et al. 1996) for performance improvement. The contour approach, on the other hand, has to rely heavily on the presence of strong object boundary. An important aspect in regional approaches is to effectively model the object appearance. The histogram approach, for instance, models the object with single or multi-model parametric modeling. Statistical modeling of appearance, however, often gives a more flexible and practical way to describe a target object. The kernel density estimation, in particular, is a nonparametric density estimation method that requires no underlying density modeling. The K-mean method, support vector machine, and relevant support machine are some examples of the kernel estimation.

The main work of this chapter is to model the object pixels in the kernel representation through different aspects and to experiment with these different
approaches. In particular, we develop the kernel-based modes for object description to facilitate tracking an object in a video sequence of moving background. Such models generally enable one to effectively handle non-rigid moving objects, account for local appearance change, and deal with partial occlusion and shade problem. We will characterize an object by its color density with implicit or explicit integration with the pixel spatial information. Three main approaches are proposed and investigated here (Huang and Jiang 2008; Huang and Jiang 2007) in this regard.

The first approach represents an object with a kernel based color density function constructed also in the context of physical locations of the object pixels. These spatial locations are derived from certain predefined basic shapes that help reflect the robustness of the object appearance. In the case of having a set of concentric annuli at the centre of the object, this concept of robustness refers to the better preservation of the appearance stability when it is closer to the object centre. The feature of a given region is represented by the color density function which is derived from the object pixels within that region. This way, the difference of the density functions across the selected regions will be able to reflect the more generic difference of the object appearance. Hence it also makes sense for the weighting factors to be introduced to signify the importance of the pixels at different physical locations when measuring the similarity of two probability density functions for the template and candidate region without the explicit use of the location parameters. Positioning an object in a newer frame is based on minimizing the distance between the density function for the object in
the previous frame and that for the candidate region. Once the centre of the object in a newer frame is located and the object shape is predicted, or rather projected there, the object shape verification process is then carried out to refine the final object border in the frame. This method is designed for tracking a non-rigid object in a video of moving background, and also for providing a flexible framework to deal with different situations arising from different types of object appearance changes, and from the different cost requirements for computation.

The second approach models the object appearance with structured samples for the kernel. The feature of the samples is represented by the density function in term of its neighbors. Thus the number of samples used to represent the object is much less than the pixels while keeping the local feature in the statistical model. Also, it is able to reduce the error in the matching steps resulting from the fixed positions of pixels since the values of the samples accommodate certain neighboring features. The localization process is similar to the first approach. It also allows weight factors to reflect the stableness of object appearance. The neighbor region of the samples can be in overlap or isolation, and the choice of the samples can be based on the characteristics of the appearance. Hence it can be used to isolate the area of larger changes from the other parts of the object. However, the size of the neighbor region, location and the number of samples need to be determined carefully otherwise it may lead to poor performance. This leads to an adaptive kernel based template that enables the object to be represented by fewer samples and thus speeds up the process of locating the object.
In the third approach, the scene is described by a kernel density estimator. The object and background pixels are modeled respectively to represent where they belong. The spatial relationships of pixels in the current frame are also modeled separately, which refer to the previous object shape and search positions. The probabilities are computed in Bayesian framework and explicitly integrated with the spatial information.

This chapter is organized as follows. We first introduce the kernel density estimation in section 2.2. Section 2.3 then presents a statistical modeling of the object appearance, incorporating the spatial relationship to strengthen the appearance stableness of the object components. Section 2.4 further proposes an adaptive kernel-based template for object tracking, while section 2.5 develops a kernel-based model of the scene for object tracking in the Bayesian framework. Experiments and some implementation issues then follow in Section 2.6.

2.2. Nonparametric density estimation

In digital multimedia, various parametric models for the signals have been used, with the Gaussian being one of the most popular models. Sometimes multiple models are needed for the target object. The target is thus ultimately described by only a few model parameters that can be estimated by such as the Expectation-Maximization algorithm. On the other hand, non-parametric estimators have also been developed as an effective tool to represent a target object. They can not only avoid the estimation on the number of models to be used and the relevant model parameters, but also have less limitation on the target the parametric models often have to impose. Hence they may achieve a
better performance at least under quite a few circumstances in terms of the accuracy of the tracking result.

The kernel density estimator \( f(x) \) is a type of non-parametric estimation for the density, often written as

\[
f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),
\]

where \( x_i \) are the samples the estimation at data point \( x \) is based on, \( n \) is the total number of samples, and \( h \) is the bandwidth. The kernel \( K \) is a symmetric nonnegative function that is centered at zero and integrates to one. It is a continuous probability density satisfying the moment conditions, \( \int K(w)dw = 1 \), \( \int wK(w)dw = 1 \), \( \int w^2K(w) > 0 \).

The bandwidth \( h \) dominates the performance of such an estimator (Wechsler et al 2004). A small value of \( h \) will lead to the undersmoothed estimation and spurious peaks in the density, while a large value of \( h \) would wipe out much of the detail in the estimated density. The \( h \) can be fixed for all samples or data points, or can be made to vary with the samples or data points. With a fixed \( h \), the estimator estimates the density at each data point \( x \) by taking the average of identically scaled kernels centered at each of the data points. The fixed-bandwidth kernel estimator is simple in implementation but may lead to a poor performance in some situations. For instance, it may smooth a part of the data while leaving other data undersmoothed or requiring a larger value of \( h \). Moreover, it may inadvertently remove many features of another part of data.
when adequately smoothing only one part of data. In fact, to better describe the actual density the $h$ should be small in areas where $f$ is large in magnitude, and $h$ should be large where $f$ is small. Also, it is often difficult to find a single bandwidth that can differentiate between distinct peaks and the valleys between the peaks for the data exhibiting multimodality. In general, the fixed bandwidth is not well suited for densities that exhibit large changes in magnitude. Hence the adaptive estimation leads to a better performance, and two of such adaptive kernel density estimations are the balloon estimator and the sample point estimator.

For the balloon estimator, a different but fixed bandwidth $h$ will be selected for each data point $x$. Thus the estimation of $f$ at $x$ is actually an average of identically scaled kernels centered at each data point. For the estimation of a data point, a new value of the bandwidth will be chosen to scale the kernels, which is able to better characterize the density. The balloon estimator reads

$$f(x) = \frac{1}{nh(x)^d} \sum_{i=1}^{n} K\left(\frac{x-x_i}{h(x)}\right),$$

where $d$ is the dimension of the space $x$, and a typical superposition into the resulting density curve $f(x)$ is illustrated in Figure 2.1.

![Figure 2.1 Estimation by a balloon estimator](image)
The sample point estimator, on the other hand, uses a different bandwidth at each sample for the data point. The estimation of \( f \) at each \( x \) is an average of differently scaled kernels centered at each data point. The sample point estimator reads

\[
f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h(x_i)^q} K \left( \frac{x - x_i}{h(x_i)} \right),
\]

and a typical superposition into the curve \( f(x) \) is illustrated in Figure 2.2.

The balloon estimator is a point-wise estimator, which can provide a straightforward analysis on the points. The drawback to balloon estimators is that when a global estimation is considered, the estimator usually fails to integrate to one due to the different smoothing parameters at each data point. However, the sample point estimator is itself a density, which is non-negative and integrates to one as long as the kernel function is also a density. The drawback for a sample point estimator is that it may suffer from a ‘non-locality’ phenomenon, where the estimation at a certain point may be strongly affected by a sample quite far away from the estimated point rather than just by those near the estimated point.

2.3. Spatial integration color model

Object appearance is often described by the histogram or intensity density and
the shape, and most object models used either color or shape description in the respective algorithms. In the following, we propose to adaptively integrate the positions of the object pixels into the adapted density representation of the object to improve the tracking performance.

2.3.1 Feature modeling in geometry

An object in a video sequence often changes its appearance to a certain extent in either color or shape, due to the variation of illumination conditions, camera viewing angle, object posture, object deformation, or to the signal resampling process. In most cases, the colors of the object pixels are correlated to the corresponding pixel positions or to the nearby positions in the subsequent frame. One color, or one group of colors, will in general not jump to a distant location suddenly. In other words, an object generally maintains certain stable appearance structured in geometry, and the changes often take place in local areas, although a part of the object may sometimes have a larger motion as in the case of a person moving his arm quickly in a video sequence. We will here mainly address the problem for the object with its main part in a smooth and slow motion. Another observation for tracking an object in a video is that the object usually has a large part in stable appearance and a small part in non-stable appearance. The stable appearance of an object is obviously the key for locating the object. Successful extraction of certain stable appearance from the object would improve the tracking accuracy while ignoring the interference of the unstable appearance. Jepson et al (2003) proposed a method that models the object stable appearance and unstable appearance by an online EM-algorithm on a long video sequence.
They decomposed the object appearance into three components: a stable component that is learned over a relatively long course, a two frame transient component and an outlier process. Their method has to be operated on certain lengthy sequences and needs a fair amount of calculation. Also, the appearance estimation error would accumulate over an increasing number of frames that could finally lead to an inaccurate approximation of object appearance. Here we will instead model the characteristics of the stableness of the object appearance in terms of geometric regions.

How can we model an object by regions so as to properly reflect its stability? Some prior knowledge is needed for this process. To start with, one should have a reasonable idea on what kind of shapes the object is likely to evolve into, and thus determine a set of basic shapes for the object to be modeled. These basic shapes may be able to sufficiently represent the potential shape changes in the subsequent frames. For example, an ellipse may be suitable for a penguin while a trapezoid can quite accurately represent a car. The parameters governing the basic shapes will be determined or adjusted throughout the whole tracking process.

By identifying stable regions of the object appearance, we can track the object by locating the more trustworthy appearance features, which may be jointly considered under a weighting mechanism. We propose to model the object by the color of the pixels as well as their spatial relationship so that they are sufficient to ensure a stable structure of the object appearance for the tracking purpose.

Since the appearance change can take place anywhere on the object, it may be
necessary to make some assumptions so that the model can be properly established and is reliable most of the time. In general, the inner region of the object bears more stable appearance, and is usually surrounded by the outer region of volatile appearance. We hence make the following regularity assumptions about the object appearance for our proposed model.

i) A moving object retains certain stable appearance within a certain period of time, e.g. within a section of a video sequence.

ii) The appearance of the central region of the object is likely to be much more stable than the regions close to the object boundary.

iii) The appearance in local object regions does not change too drastically. In other words, the shape change or transition is somewhat smooth. Otherwise a prior knowledge for a sharp change is available.

iv) The geometric centre of the object remains on the object, similar to a convex shape.

We assume that the scale of admissible appearance change is relatively uniform within each region and, in any case, the object is able to be partitioned to fulfill this assumption as much as possible. Since the object may turn towards different directions as it moves across different frames, it would be beneficial to select a base shape so that it remains invariant when the object rotates around the centre.

For the representation of an object in our model, under the above assumption for the appearance stableness, we define a set of annular regions $S_k$ by the
concentric circles at the object centre, as shown in Figure 2.3 (a), where the object template is represented by the area within the dashed contour. The $S_k$ are sequenced in such a way that $S_1$ is the disk containing the centre, and $S_{k+1}$ is the annulus next to $S_k$ on the outside. The pixels on the object are then grouped for each annulus $S_k$, and the object is represented by these groups of pixels in the annular regions. This way, the characteristics of the pixels in these groups will constitute part of the object features.

Though the characteristic of a group of pixels can be represented by different models via such as the mean value or histogram, we propose here a statistical representation, in the form of kernel density estimation, to characterize the group of pixels. In other words, the object feature is characterized by a probability density of the object pixels falling into each annulus, which can be calculated by a kernel density estimator on the pixel color within the annulus, as shown in Figure 2.3 (b).

![Figure 2.3](image)

**Figure 2.3** (a) Annular regions $S_k$. (b) Region-wise color density on $k$.

Let $r_0=0$, and $r_k$ for $k=1, \ldots, m$ be the increasing radii for the annuli. An annulus $S_k = \{ x: r_{k-1} \leq \|x-s\| < r_k \}$, where $x$ and $s$ are both two dimensional vectors and $s$ denotes the centre, is thus represented by the positions of all the pixels on the
annulus or disk when $k=1$. For each pixel position $x$, we denote by $I^o(x)$ the colors at the pixel in the object template in the previous frame. We note that this symbol actually denotes a color vector and that such a vector representation will also be adopted throughout the thesis. We further let $R_k$ be the object region within $S_k$, i.e. $R_k = \{x \in S_k: x \text{ is on the object}\}$, then the object can be represented by the probability distribution of the object pixels in the annuli. To obtain the probability distribution we use the nonparametric density estimation which can be expressed as

$$p(k,s) = \frac{1}{C(s)} \sum_{x \in R_k} \phi(|I^o(x-s)|),$$  \hspace{1cm} (2.1)

$$C(s) = \sum_{x \in R} \phi(|I^o(x-s)|),$$

where $R = \bigcup_k R_k$, $C$ is a normalization factor, and $\phi(\cdot)$ is a kernel function such as the Gaussian, the triangular, or the bi-weight functions. Such a density distribution, as illustrated in Figure 2.3 (b), implicitly characterizes the object appearance by mean of the pixels annularly distributed relative to the object centre. For simplicity, we often drop the parameter $s$ in $p(k, s)$ and $C(s)$ when there can be no confusion. Hence, the density $p(k)$ for a smaller $k$ reflects the characteristics of the pixels close to the object centre and the $p(k)$ for a larger $k$ reflects the characteristics of the pixels relatively far away from the object centre. With this representation, we will then be able to analyze and process the pixels with the implicit consideration of their physical locations.

For a candidate region in the current frame of timestamp $t$, we denote by $s'$ the derived object centre $s$ at timestamp $t$, and denote by $I(y)$ the color values at the
pixel position $y$. Then the density for the annulus in the current frame, similar to (2.1), is

$$q(k, s') = \frac{1}{C(s')} \sum_{y \in R_k} \phi(| I(y - s') |),$$

where $R_k$ is the candidate object region in the current frame. The density $q(k)$ is similar to other types of color densities that can statistically represent the object features except that ours here are constrained to the pixel locations.

There are several advantages of this representation for the object.

i) It integrates the pixels’ location into the color description without the explicit use of any location parameters, thus simplifying the later matching steps.

ii) The geometric base shape for the region division can vary, and is object dependent. One can choose a proper geometric shape that can best reflect the change distribution of the object appearance.

iii) The statistical representation of pixels in a region reduces the effect of changes on the number of object pixels in that region. It is thus able to better retain the object characteristics.

iv) The probability density proposed in this method provides a straightforward connection between the pixel locations and the pixel colors. Since the pixels of similar appearance and importance are largely grouped together within a base shape, such importance can be easily strengthened or weakened by selecting proper coupling weights.
v) It facilitates the coarse-to-fine searching strategy in the later matching process.

2.3.2 Measurement metric

If we use the object density \( p(k) \) at time \( t-1 \) as the template, and compare it with the density \( q(k) \) of a candidate region in the current frame of time \( t \), then the proper position of the detected object will be obtained by minimizing the difference of the densities. Further, since the density over \( k \) relates to the stableness of appearance, the closeness of these two density functions should be weighted differently for the different elements derived from the different annuli.

For this purpose, we introduce a coupling weight function \( \varphi(r) > 0 \) that is monotonically decreasing with \( \varphi(r) \to 0 \) as \( r \to \infty \). For any two distributions \( p(k) \) and \( q(k) \), we define their inner product as \( (p, q) = \sum_{1 \leq k \leq m} \varphi(r_k) p(k) q(k) \). Then these two distributions, or vectors, are considered more similar to one another, the more \( (p, q)^2 / [(p, p)(q, q)] \) is closer to 1. The weight function can give more significance or trust to the region closer to the object centre. This is consistent with the fact that a tracked object is more likely to deform along its boundary than in the centre area. Hence our model of annuli is particularly ideal to accommodate the changes that are somewhat proportional to the distances to the centre.

The object location in the current frame should then be the one that minimizes the distance between the density function for the object template and the density function for the candidate region. The distance metric for the two density functions could simply be a Euclidean distance. However, in order to
accommodate the potential illumination changes in the frames, which is a common phenomenon in video sequences, we calculate the $s'$ via

$$ s' = \arg \min (1 - \omega(s, s'^{-1})) $$

$$ \omega(s, s'^{-1}) = \frac{\sum_{k=1}^{m} \phi(r_k) \hat{q}(k, s) \hat{p}(k))}{\sqrt{\left( \sum_{k=1}^{m} \phi(r_k) \hat{q}^2(k, s) \right) \left( \sum_{k=1}^{m} \phi(r_k) \hat{p}^2(k) \right)}} $$

$$ \hat{q} = q - \bar{q}, \quad \hat{p} = p - \bar{p}, $$

where $\bar{q}$ and $\bar{p}$ are the mean value of density for time $t$ and time $t-1$ respectively.

This calculation in (2.3) corresponds to the use of the following similarity metric $\hat{\partial}(p, q)$

$$ \hat{\partial}(p, q) = 1 - \frac{\langle p, q \rangle}{\sqrt{\langle p, p \rangle \cdot \langle q, q \rangle}} $$

$$ p = (p(1), ..., p(m)), \quad q = (q(1), ..., q(m)) $$

and such a metric will be able to preserve the linear transform of the color brightness, e.g. $I(y) = aI'(y) + \beta$.

A brute force search for $s'$ via (2.3) could be quite inefficient. We hence make use of the 2-D-log search method (Jain and Jain 1981) developed under a different setting. The search in the current frame starts from the object centre position of the previous frame, or from the projected centre position estimated from the previous centre moving speed. Each step tests five points in a diamond arrangement and repeats the diamond search in the next step with the centre moved to the best matched point. Nine search points are examined at the last step when the step size is reduced to 1 pixel.

To further reduce the computational load, several strategies can be considered.

1) Adopt fewer annuli for the object model without decreasing the accuracy
significantly. 2) The search for the object centre can be conducted in two, coarse to fine, steps. The first step is to find a suitable candidate location using a smaller \( m \), such as \( m=3 \), in (2.3). The second step is to further check the total difference of two densities between the candidate region and the object template. 3) The object pixels for inspection can be sub-sampled. For instance, one pixel can be selected from every pair of consecutive pixels in space, resulting in a half number of pixels for the model. 4) The searching can be made more directional if the object motion history is available. We also note that we can adjust the representation scale of the object with the number of annuli involved.

2.3.3 Shape verification in the outer region

Once the centre location of the object is determined, we can proceed to refine the object in the current frame if it is a non-rigid object. Most of the work would be to determine weather the pixels near the object boundary remain in or outside the object due to the shape deformation during the object motion. Some pixels belonging to the object in the previous frame may have been excluded in this extracted region, and some pixels in the extracted region may no longer be part of the object. The verification of outer pixels here is to examine the pixels near the predicted object boundary, i.e. the previous object boundary projected into the current frame based on the predicted location from the previous frames. The area for the verification is the area within the distance \( \varepsilon \) to the candidate object boundary in the current frame, as shown in Figure 2.4, where the area \( \Omega_c \) is between \( \Gamma^+ \) and \( \Gamma^- \). \( \Gamma^t \) is the predicted candidate object border, and \( \Gamma^{t-1} \) denotes the corresponding object border in the previous frame. The search area for an
examined pixel $I(y)$ is the candidate object region within the distance $\iota^s$ to the examined point. The area $\Omega_s$ marked with the dash lines in Figure 2.4, and the pixel color should be the object template color $I'(x-d)$ in the previous frame rather than the object color $I(x)$. In other words, if we want to determine whether a pixel $I(y)$ near the candidate boundary actually belongs to the object or not, we compare it with those object pixels $I'(x+d)$ for $d=s^{t-1}$ and $\|x-y\|<\iota^s$. To improve the matching accuracy, the neighboring pixels are taken into consideration for comparison. We thus design a neighborhood mask $M$ for selecting the neighboring pixels (see Figure 2.5 for an example). Then the object pixel that has the minimum difference with the examined pixel in term of their respective neighbors can be found within the local search area, and the subsequent difference $\Delta$ is obtained for the examined pixel to be considered if it is likely to belong to the object. This difference $\Delta$ is expressed as

$$\Delta = \| I(y) - I^o(x_{\text{min}} - d) \|,$$

$$x_{\text{min}} = \arg\min_{y\in\Omega_s, z\in M, x\in\Omega_s} \sum || I(y + z) - I^o(x + z - d) ||,$$

where the spatial variable $z$ (distance from a neighbor pixel to $x$ or $y$) goes over all the selected neighboring locations, and $\arg\min$ refers to an operation which returns the argument value ($z$ in this case) that minimizes the operand the operator is applied to.

We note that Figure 2.4 illustrates the verification of pixels on whether they remain on the object. The area on the right of solid curve $\Gamma^\iota$ is the candidate
object region obtained via (2.3), the area $\Omega_e$ between the round-dotted lines needs to be examined, the slash dash line is the search region $\Omega_s$ for maximum similarity of object pixels, and the dash-dot contour demarcates the object’s original location but in the current frame. Finally, the verified object areas are extracted via a suitable threshold on the difference $\Delta$ along the object boundary. The threshold can be determined so that the number of pixels that are less than the threshold in the outer area is approximately similar to that in the area between the projected contour and its outer line at the distance $t^v$.

![Figure 2.4 Pixel-wise verification](image)

![Figure 2.5 Neighborhood mask](image)

The accuracy of the above verification is affected by the selected parameters $t^v$ and $t^s$, as well as the pixel neighborhood mask $M$. If the object and its surrounding background are well distinguished, then it allows a large value of $t^v$ to verify the object of large deformation. Otherwise a smaller $t^v$ would lead to
less interference from the non-object background. Similarly, \( \tau \) defines the search area where a large distance is better suited for a large object deformation, while a small distance may leave a part of object out of the detected object region.

2.3.4 Model parameters and their impact on performance

The performance of this approach of nonparametric density estimation will be affected by the chosen model parameters. They include the number of annuli, the radius, the basic shape, the search region, the size of the neighborhood mask, weighting factors and the kernel bandwidth.

We recall that the basic shape should best reflect the shape of the object of interest, and a disk shape is ideal to accommodate the rotation invariant features. In some cases, more than one shape may be suitable to describe the object. The choice of the number of annuli depends on the size of the object to be tracked. Since the local appearance of the object is modeled by the kernel statistically, the number of pixels in a local region should not be too small otherwise the density obtained will not be able to characterize the local region and will result in poorer performance in locating the object. On the other hand, increasing the number of annuli also increases the computational cost in the appearance modeling step as well as the matching step. The variation of the annuli radius is hence allowed. For example, the radius of an annulus closer to the object center can be larger than that far away from the center, leading to a similar number of pixels in each annulus.

The kernel bandwidth \( h \) reflects the extent the pixel values contributed locally
to the density. Though the accuracy of density to be estimated is not the main concern in this approach, because of the same effects on both the template and the candidate region, the kernel bandwidth will inevitably affect the later matching process in terms of whether it can properly represent the features of the density. For instance, a larger bandwidth would lead to a smoother density and wipe out some feature details. This would thus decrease the feature differentiation on the densities being compared. Though a smaller bandwidth seems better in characterizing the annular features and leads to a better matching later on, a narrow bandwidth would diminish the contribution of pixels of larger intensity values. If most pixels in an annulus have large values, then the obtained density there does not properly represent its pixels feature. Then the comparison actually takes place by only taking those pixels of small values. In some cases, this would not be accurate enough to locate the object.

In the outer region pixel verification, the neighborhood mask decides the amount of neighboring pixels contributing to the verification. In general too large a neighborhood mask not only increases the computational cost but also may result in less accuracy near the object boundary since they are partly mixed with the background.

The region to be verified for the object boundary is estimated on the extent the object may deform. If it is too small, then the verification will not cover the part of the object that falls out the region. If it is larger than necessary, it may then lead to some background noise into the object, especially when there is a similar background existing near the object. Moreover, a larger region for verification
increases the computational cost.

In the steps of locating the object, the weight factors that balance the stableness of appearance are determined based on the prior knowledge at the starting frame. In the subsequent frames the weight factors should be adjusted automatically based on the previous results. In other words, we compare the obtained object in the current frame with that in the previous frame. If the density of a region has undergone a larger change, then this region tends to be unstable, and the weight factors will be smaller for the same process in the next frame.

The approach needs the prior knowledge about the object shape and trend in appearance change. If the appearance changes abruptly in the sequence on occasions but the shape is not adjusted accordingly, then the performance may suffer. We also note that we may relax on the assumption that the geometric centre of the object remains on the object, as long as the hollow area (background pixels) in the inner region can be modeled with a proper shape and it would not change dramatically in the video sequence.

2.4. Adaptive kernel template

When the object appearance changes locally, there is an alternate and better way to model the object, allowing a suitable object representation that impacts little under such changes. It is similar to the traditional template approach but the pixels are actually represented statistically and they characterize the local group of the pixels. We derive the model in the following.
2.4.1 Motion estimation

Let $I(x)$ and $P(x)$ be the intensity vector of a pixel at position $x$ in the current frame and the previous frame respectively, or referred to as $I$-frame and $P$-frame for short. In general, the displacement vector $d$ of the object in the two consecutive frames can be estimated by the minimum error between the object pixels in the previous frame and candidate pixels in the current frame, i.e.

$$d = \arg \min \left( \sum_{x \in R^o} \| P(x) - I(x + d') \| \right)$$  \hspace{1cm} (2.4)

where $R^o$ is the object region in the $P$-frame, and the arg is with respect to $d'$, the object displacement of the frames. This $d$ is known as the Displaced Frame Difference (Wang et al 2002), and one thus locates the object in the current frame by minimizing the difference between the object template in the previous frame and the candidate region in the current frame. The difference is calculated by comparing each pixel on the object template in the previous frame with the pixel corresponding exactly to the relevant position in the candidate region in the current frame. Since the pixels in the current frame may deviate somewhat from their corresponding positions in the previous frame due to such as illumination changes, local motion, object deformation, or the sampling or quantization process applied in the current frame, to accommodate such object variations, one may modify (2.4) into

$$d = \arg \min \left( \sum_{x \in R^o} \min_{l \leq L} \left( \| P(x) - I(x + d' + l) \| \right) \right),$$  \hspace{1cm} (2.5)

where vector $l \geq 0$ is a local step away from the location $x + d$ for the search, and $L$
is the maximum search distance. This means that each object pixel in the $P$-frame finds a matching one in the $I$-frame according to the minimum difference in the pixel’s local area rather than just with the pixel at the matching position. If one adopts further the concept of pixel blocks in the similarity computation, then formula (2.5) becomes:

$$d = \arg \min_{x \in 
\bar{b}, \ y \in \Omega} \min_{y \in \Omega} || P(x + y) - I(x + y + d + l') ||,$$

where $\Omega$ represents a pixel block centered at $x + d + l$, and $y$ is the radius of the block. This is to search for the most similar pixel in the corresponding pixel neighborhoods (see Figure 2.6 (b)) hence each object pixel at location $x$ in the $P$-frame is compared blockwise via $\Omega$ with those within the circled area centered at its corresponding location $x + d + l$. The dotted line marks the object outline, the dash dot line indicates the block $\Omega$ for the block match, and the long dash double dot line demarcates the local search area with radius $L$.

![Figure 2.6](image)

**Figure 2.6** (a) Object in previous frame. (b) The candidate region in the current frame.

Although this method will improve the accuracy of the estimation of the object motion by taking into account the local changes, it is computationally very
expensive. To overcome this drawback, we approach the problem in a different way to avoid using nearby pixels directly in each comparison. We will define for a given pixel a statistical feature based on the neighbors of the pixel, and use this newly defined feature to represent this neighborhood. As a result, we can use such feature values for far fewer pixels to represent the object, and eventually simplify the calculations. The collection of these selected pixel positions and their respective feature values will form a kernel-based template. If an object is partitioned into a set of pixel blocks, we define for the center pixel $\xi$ of each of these blocks

$$\psi(\xi) = \frac{1}{C} \sum_{\eta \in \Omega} \phi(||P(\eta)||),$$

$$C = \sum_{\xi \in \Xi} \sum_{\eta \in \Omega} \phi(||P(\eta)||),$$

where $\Omega_\xi$ is the block centered at $\xi$, $\Xi$ is the set of all such sample positions $\xi$, $\phi$ is a kernel function, and the kernel element $\psi(\xi)$ is the newly defined feature at $\xi$. Since $\psi(\xi)$ each characterizes a statistical view of the neighborhood pixels around $\xi$, they together constitute a simplified representation of the object in the $P$-frame.

Similarly, the current frame $I$ can also be represented by the kernel elements via (2.7) with the $P(\eta)$ and $\psi(\xi)$ there being replaced by $I(\eta)$ and $\Psi(\xi)$ respectively, whereas other parameters such as the block size and bandwidth remain the same as the object template. The kernel elements $\Psi(\xi)$ for the current frame can be calculated all at the same time before any of their use, or calculated
only for the corresponding candidate kernel elements in each matching step. However, if the pixel blocks vary in size for the object template, then the latter approach has to be used since it would be too complicated to assign multiple values to a kernel element in the former. If many search steps are needed to locate the object position, the latter may incur more computational cost.

The kernel based template simplifies the object representation with fewer sample elements, and thus reduces the computational cost required for locating the object. The object motion can then be estimated according to the maximum similarity of the kernel elements between the given object template and the candidate region in the current frame

$$
    d = \arg \min_{\xi \in \Xi} \min_{l \leq L} \| \psi(\xi) - \Psi(\xi + d' + l) \|.
$$

(2.8)

The main difference of this function from (2.5) or (2.6) is that it only compares a few kernel elements in the matching process. If all the blocks have the same size, then these feature values can be calculated once at the current frame. The searching path would thus not increase the computational cost by much, whereas algorithms based on (2.5) or (2.6) must compare all object pixels along the searching path. Furthermore, it is possible to introduce weight factors in the matching step so as to better control the appearance stability. The $L$ can be made small or totally ignored in (2.8) since the kernel elements already accommodate the neighborhood characteristics.
2.4.2 Adapted elements

A kernel element is able to feature the pixel and its surrounding pixels. A simple way to obtain the layout or structure for kernel elements is to divide the object into a set of blocks with the same or varied size and these blocks are fully within the object area. That is, for the kernel elements $\psi(\xi)$ at $\xi$, we have $\xi \in \mathbb{R}^o$ and $\Omega_\xi \in \mathbb{R}^o$, where $\Omega_\xi$ implicitly depends on the block size $r$.

The better kernel elements are those that are obtained for the more meaningful pixels with such as uniform color, regular texture, discriminative color from the background or from regions with similar stability. The kernel elements, along with their location structure, will be termed as an adaptive kernel template. In the searching process, the same structure and kernel blocks are applied to the current frame. The structure may be slightly adjusted in the matching step, as illustrated in Figure 2.7 (b), due to the potential local location shift of the feature points. This approach is also less susceptible to the object deformation or illumination changes since the element representation has already accommodated such noises. Figure 2.7 (a) illustrates a simple element layout.

![Figure 2.7](image)

Figure 2.7 (a) Object structured kernel elements. (b) Search similar structured kernel elements in the current frame.
The number of elements selected would be crucial for the performance. In other words, the extraction of proper kernel elements is important. The blocks for the elements can overlap or be distant to one another, and they do not have to form a regular shape, depending on the appearance features and their locations. This adapted template is more flexible than the approach previously described. For instance, a kernel element can be shifted to reflect the shape changes if a prior information about a likely specific deformation is known. A very small number of kernel elements will obviously further reduce the computational cost, but they may not be able to sufficiently represent the underlying object features and finally lead to the tracking failure. If the block size is too large, or the bandwidth in the kernel function is too large, then the derived elements may lose their unique characteristics. On the other hand, too small a block size or bandwidth could again break the representativeness of the local regions. For more meaningful kernel elements, one may choose to apply segmentation to the object to help to achieve this goal. In chapter 4 later on, we will develop another method with a similar concept.

2.4.3 Deformable object

We recall that the object motion is estimated by minimizing the errors of kernel elements between the object template and the candidate region in the current frame. If the object is non-rigid or deforms in shape due to changing the viewing angle, the adaptive kernel template is still able to achieve a satisfactory result by properly adjusting the structure of the kernel elements as mentioned above. Since all the elements contribute similarly in the adaptive template match
in (2.8), we may even adjust the elements’ significance by multiplying weight factors to the kernel elements, resulting in

$$d = \arg\min_{\xi \in \Xi} \sum_{\xi \in \Xi} \lambda_{\xi} \min_{l \leq L} \| \psi(\xi') - \Psi(\xi + d' + l) \|$$

(2.9)

where \( \sum_{\xi \in \Xi} \lambda_{\xi} = 1 \). For instance, it is acceptable to apply a relatively small weight factor for the outer kernel elements near the object boundary in the structured representation. Moreover, smaller weight factors can be utilized to reduce the effect of unstable regions in the interior area of the object. Prior knowledge about the appearance stableness is needed to decide the \( \lambda_{\xi} \) at the starting frame. After that the process will automatically adjust the weight factors according to the observation on the past frames. We can also use a larger block size near the object boundary compared with that for the interior, implying the representation has fewer significant features over this area, or use a larger bandwidth in the feature calculation of the kernel elements in the outer or unstable region of the object since a larger bandwidth smoothes the underlying feature and the integration of all the probabilities to one is not required here.

If a more accurate object boundary is required for such as the need to update the template for the next frame estimation, we can apply the outer region verification method proposed in the section 2.3.4 for this purpose. For the completeness of our procedure, we summarize it in Procedure 2.1.

### 2.5. Kernel-based pixel modeling

The probabilities of pixels in the current frame belonging to the object or
background can be estimated by comparing the similarity of pixels with the pixels in the corresponding region in the previous frame. Here we introduce the scene modeling with kernel-based estimation. The likelihoods of pixels are then incorporated into a Bayesian posterior framework. With the help of pixel spatial modeling for the object pixel cohesion, the movement of the object can be estimated by maximizing the probability sum based on the total probabilities of the posterior probability and the spatial probability.

2.5.1 Scene modeling

We will model the scene with its color and spatial relationship respectively, rather than only focus on one or try to combine them at the same time as in (Sheikh and Shah 2005), which resulted in high dimensional computation. When we model the scene color, the spatial information can be neglected, so that it allows us to describe the scene with pixel color density. When we model the pixels’ spatial relationship, we describe the pixel with the spatial distance to a reference location, typically the estimated centre of the object. We generally assume that the object is a solid shape without holes in the interior area, and the shape of the object can be approximated by certain basic shapes such as ellipses. We will use a non-parametric density estimation method to derive the color densities of the object and the background in the current frame, where the object is what we need to locate. The object and background in the previous frame are known prior to this approach. In the following, we describe the methods for modeling the color and spatial information of the object and the background respectively.
The pixels in the current frame are represented as $I(x)$, where $I$ is the color vector, and $x$ represents the location of the pixel. The object pixels in the previous frame are given by $R^o = \{ P_k \}$, $k=1...m$, $P$ is the color vector, and $m$ is the number of object pixels. Then the likelihood of the pixels in the current frame
belonging to the object, \( p(I|R^o) \), can be represented by the probability of kernel function centering at \( I(x) \). This is expressed as

\[
p(I \mid R^o) = \frac{1}{m} \sum_{k=1}^{m} \phi\left( \| \frac{I - I_k^o}{h} \| \right),
\]

where \( \phi \) is a kernel function, \( h \) is the kernel bandwidth.

Similarly, the background pixels in the previous frame are given by \( R^B = \{I_k^B\} \), \( k=1\ldots n \), \( I_k^B \) is the color vector and \( n \) is the number of pixel in the local background \( B \). The likelihood of the pixels in the current frame belonging to the background, \( p(I|R^B) \), then read

\[
p(I \mid R^B) = \frac{1}{n} \sum_{k=1}^{n} \phi\left( \| \frac{I - I_k^B}{h} \| \right),
\]

where \( \phi \) is a kernel function as in (2.10). If a fixed bandwidth \( h \) is used and \( \phi \) is a density, then the likelihood \( p(I|R^o) \) and \( p(I|R^B) \) are both densities. The bandwidth \( h \) will determine the characteristics of the estimated densities based on the known object or background pixels. If we wish to weaken the effect of intensity far away from the estimated pixel, we may choose a smaller bandwidth. Although a variable bandwidth could better characterize the underlying densities as mentioned in the last section, different bandwidths would have to be carefully determined.

### 2.5.1.2 Spatial modeling

For modeling the pixels’ spatial relationship, we measure the spatial information with respect to the estimated object center \( s \). It is assumed that the
pixels closer to the object center will have higher probabilities to remain on the object, and the pixels far away from the center are more likely to become the background in the current frame. Hence we hereby model the spatial relationship of the object pixels by \( p(x(s)) \) with respect to the distance from the pixel location \( x \) to the estimated object center \( s \), which is expressed as

\[
p(x(s)) = \varphi\left(\frac{x - s}{\sigma}\right)
\]

(2.12)

where \( \sigma \) is the bandwidth, and \( \varphi(r) \) is a monotonically decreasing function such that, \( \varphi \rightarrow 0 \) as \( r \rightarrow \infty \), and \( \varphi(0) = 1 \). For better illustration of object pixels’ spatial distribution, the \( \sigma \) can adapt to the object shape known in the previous frame. If the modeling function is Gaussian, the standard deviation \( \sigma \) would vary with the shape of the object. The problem is simplified if the object shape can be emulated by a regular shape or multiple regular shapes. For instance, if an ellipse can be used to represent the object shape, the standard deviation of the Gaussian can be easily obtained based on previous object shape. If we use the pole coordinates \( \rho \) and \( \theta \) to represent the pixel location \( x(\rho, \theta) \) with respect to the object center, then the standard deviation \( \sigma \) is simply

\[
\sigma = \frac{ab}{\sqrt{b^2 \sin^2(\theta - \vartheta) + a^2 \cos^2(\theta - \vartheta)}},
\]

(2.13)

where \( a \) and \( b \) represent the long diameter and short diameter of ellipse respectively, and \( \vartheta \) is the rotation angle of the ellipse, as shown in Figure 2.8.
If the object cannot be well described by one regular shape, multiple shapes can be utilized. The largest shape is the main one and the others are considered local to the main, which means the $p(x)$ of a pixel in this local area is calculated with the local shape rather than with the main shape. For example, if a second ellipse centers at $s'$ with long diameter $a'$, short diameter $b'$ and rotation angle $\vartheta'$, then its standard deviation $\sigma'$ will be calculated via

$$\sigma' = \frac{a'b'}{\sqrt{b^2 \sin^2(\theta - \vartheta) + a^2 \cos^2(\theta - \vartheta')}}.$$

For the consistency of the two shapes, the spatial probabilities are set to

$$p(x) = \frac{1}{\alpha \sqrt{2\pi}} e^{-\frac{||x-s'||^2}{2\alpha^2}}$$

for $||x-s'|| > \sigma'$, and

$$p(x) = \frac{1}{a \sqrt{2\pi}} e^{-\frac{||x-s'||^2}{2\sigma'^2}}$$

for $||x-s'|| \leq \sigma'$, where the parameter $\alpha$ may be adjusted. The scaling effect for multiple shapes model is illustrated for example in Figure 2.9 with $\alpha = a$. The
spatial probabilities in Figure 2.9 (b) before scaling do not well reflect the spatial probabilities of object pixels relating to the object center under the pre-assumption that a pixel closer to center in distance is more likely to be an object pixel. On the other hand, the probabilities in Figure 2.9 (a) after scaling can represent the object region and the main shape is the dominant area.

The $p(x)$ represents the spatial probability of the pixel with respect to the hypothesized object center. In other words, it implies the cohesion of object pixels. The pixel with higher probabilities is more likely to be an object pixel, and the main shape becomes the dominant area. The spatial probabilities are there to help extract suitable pixels for the object, and to eliminate the background pixels which exhibit similar color to the object and lead falsely to high posterior probabilities.

![Figure 2.9](image)

**Figure 2.9** (a) Spatial probabilities after scaling. (b) Before scaling. (c) Original image.

### 2.5.2 Object extraction

We already have a color model and a spatial model for the object and background respectively. In the next step we put it into the Bayesian framework to obtain the probability of a pixel belonging to the object region. The likelihood of being an object pixel and background pixel is represented by these densities.
from the kernel density estimation, the prior probabilities of object pixels and background pixels are assumed to be the same, and the posterior probability is calculated via

$$
p(R^o \mid I) = \frac{p(I \mid R^o)p(R^o)}{p(I \mid R^o)p(R^o) + p(I \mid R^b)p(R^b)}.
$$

This probability basically illustrates how likely a given pixel of color $I$ belongs to the object in the current frame. In other words, a given pixel being an object pixel depends on the color similarity. Hence the pixels with high probabilities can be extracted by a threshold or in terms of neighbor probabilities. Next we incorporate the spatial relationship into the resulting threshold probabilities, where the object pixels closer to the object center are enhanced and background pixels away from center are weakened. Since the spatial relationship depends on the assumed object center, the object location should attain the maximum sum of total probabilities from color probabilities and spatial probabilities, and is expressed as

$$d = \arg \max \sum_{x \in R} p(R^o \mid I)p(x(s)),$$

where $R$ is the whole frame region, and $d$ is the replacement of object. If we denote the object center in the previous frame as $s^o$, then (2.14) can be rewritten with (2.12) as

$$d = \arg \max \sum_{x \in R} p(R^o \mid I)\phi(\frac{x - \|s^o + d\|}{\sigma})$$

where the current center $s = s^o + d$. The new object location is determined by the
sum of all pixels’ probabilities combining the color similarity and spatial probabilities.

In the case of the spatial model not being sufficient to represent the full object area, and there may be nearby background pixels exhibiting colors similar to the object, the sum of probabilities can be based on the previous object shape. In other words, the previous shape is projected onto the current search location and only those probabilities within the shape are summed. Thus the (2.15) can be further rewritten as

$$ d = \arg \max_{x \in R_s} \sum p(R^o \mid I) \phi\left(\frac{x - \| s^o + d' \|}{\sigma}\right), $$

(2.16)

where $R_s$ represents the area within the object shape in the current search location centered at $s^o + d'$.

In order to reduce the computational load, the search for the new object center should start from the object center in the previous frame for a slowly moving object. If the moving trend can be estimated from the historical frames, the starting point for the search can be reassigned based on the object’s historical speed and direction. Once the displacement is obtained, the object can be extracted and refined if necessary as in section 2.33. The extracted object is then used to update the simulating shape for the object in the next frame. The main steps are summarized in Procedure 2.2.

The drawback of this approach is that the spatial relationship is based on the estimated shape from the previous frame. If the error of extracted area
accumulates along historical frames and the shape is not able to be adjusted in time, the object location may deviate from the correct position. Thus it may be less suitable for fast-moving and largely deforming object unless there is no interference from its local background.

2.6. **Implementation and Experiments**

The proposed models have advantages to better handle certain circumstances in tracking an object in video sequence. We demonstrate their performance in the
following experiments.

2.6.1 Experiments on spatial integration color model

The video frames for experiments are shown in Figure 2.10 (a) and Figure 2.13. The frame size is 240x320 pixels. For these experiments, we can choose for the kernel function $\phi(\cdot)$ in (2.1), (2.2) and the weight function $\varphi(\cdot)$ in (2.3) the following $N$-dimensional Gaussian

$$
\psi(z) = \frac{1}{(\sqrt{2\pi}\sigma)^N} e^{-\frac{zz'}{2\sigma^2}}
$$

or the triangular function $\psi(z)=1-||z||$ for $||z||\leq1$ and $=0$ if otherwise, where $z$ is an $N$-dimensional row vector and $z'$ is its transposition. The Gaussian is traditionally a popular kernel function in the literature. We note that the function for object template and weight can be similar or different, and the standard deviation $\sigma$ should remain the same for both the object template and the candidate region. The object location is thus determined by minimizing the error between the template $p(k)$ and the candidate region $q(k)$ via (2.3).

We now illustrate our experiments on two video sequences. For the first sequence, we took $m = 8$ for the object model, and set the radius $r_k = 5k+1$ pixels for the concentric circles. The density function for the object is obtained via (2.1). Then we applied the fast search algorithm (Jain and Jain 1981) to locate the new position of the object which has the most similar density to the object model via (2.3). The result is shown in Figure 2.10 (b), where the brighter (green) boundary indicates the new location while another boundary depicts the original
object shape in the previous frame as in Figure 2.10 (a). Next we verified or refined the object shape. The local area to be considered for shape deformation was chosen to be within the distance of $\rho^v < 8$ pixels along the projected boundary for the detected candidate object. The mask for selected neighbors was a diamond shape of thirteen pixels in total as shown in Figure 2.5.

The resulting object contour and the final fully extracted object are shown in Figure 2.10 (c) and (d). The brighter (green) contour in (d) can be regarded as refined or adjusted from the original one in (a). In this shape refinement step, our experiments show that the determination of the pixels on the upper sides of the object is not so effective due to the background being somewhat similar to the upper part of that object. If the object shape deformation is not expected to be large, then the refinement operation can be restricted to a narrower area corresponding to a smaller $\rho^v$ in Figure 2.4. Typical frames with extracted object contours in the sequences are also depicted in Figure 2.11 below, where the contours are again shown in a bright (green) color.

There can be various different similarity or distance measures, like that in (2.3), which can potentially affect the tracking performance. We thus conducted in Figure 2.12 a comparison by different similarity measures. Figure 2.12 (a), (c), (e) are based on the standard Euclidean distance and Figure 2.12 (b), (d), (f) are based on our proposed measure in (2.3). In Figure 2.12 (a) and (b), both distance measures lead to similarly correct object location in the subsequent frame. However, when the frames are preprocessed to reflect the potential illumination changes by undergoing a certain degree of linear transformation, the results
due to the different measures are significantly different as in Figure 2.12 (c) and (d), where latter corresponds to the metric in (2.3). The experiments showed that the similarity measure (2.3) can still locate the object properly in (d) whereas the Euclidean distance has led to a significant deviation from its correct position as shown in (c). Further experiments were even conducted on some simple nonlinear transformations. In fact Figure 2.12 (e) and (f), also for the Euclidean distance and our proposed similarity metric (2.3) respectively, illustrate the tracking
results when the color has undergone a quadratic transformation. The latter again holds out well as seen in Figure 2.12 (f) while the former falters as expected. This empirical result for the nonlinear transformation is somehow unexpected, and may be somewhat accidental for this case. In any case, the similarity metric in (2.3) is more robust at handling the tracking, especially when there are lighting changes or the like in the scene.

Figure 2.12  Comparison on locating the object with two different similarity measures

Figure 2.13  Parrot movement and the traces of its centre
For a different setting, we conducted another experiment on a parrot video sequence, and the results are summarized in Figure 2.13. To highlight the object traces and avoid the cluttering of the contours, we only plotted with a bright dot the centre of the object parrot. The video sequence starts from the frame in Figure 2.13 (a), the bright heavy dots will each denote the object centre of a particular frame. Figure 2.13 (b), (c), (d) and (e) thus depict the traces of the parrot across a consecutive sequence of frames. In creating the last picture (f) of Figure 2.13, the parrot template has actually been adjusted during the frame by frame object tracking since the appearance of the parrot has undergone relatively larger changes.

These experiments have thus demonstrated the success of the proposed approach in capturing the object in the subsequent frames.

2.6.2 Experiments on adaptive kernel template

For the kernel-based template approach, we first examined the effect of block size for the performance. The video sequence used is again that in Figure 2.10 (a) for an easy comparison. We first set the block size and selected the block locations, as in Figure 2.14 (a), for calculating the kernel elements in the first frame. The shape of blocks can be set to such as disk, square or polygon, and the blocks can be adjacent, overlapping, or quite apart from each other, as long as all of them are completely within the object region. The kernel function $\phi$ used for the kernel elements is a Gaussian function. The locations at which the kernel elements are calculated via (2.7) are shown in Figure 2.14 (b) whose blocks of 9x9 pixels are shown in Figure 2.14 (a). For simplicity, we here arranged the
blocks in a regular manner, and it is generic anyway with no large impact on the actual performance.

![Figure 2.14](image)

**Figure 2.14** (a) Blocks. (b) Locations of kernel elements. (c) Object location.

With the kernel-based template in Figure 2.14 (b), the object formation in the subsequent frame is conducted via (2.9). The derived object is outlined in Figure 2.14 (c), which actually depicts the 22\textsuperscript{nd} frame after the one in Figure 2.14 (a) since the object moves slowly during this time slot. It shows the location is quite accurate despite the unrefined deforming boundary. The effect of block size on the kernel-based templates is shown in Figure 2.15 for three different cases.

![Figure 2.15](image)

**Figure 2.15** Kernel elements. (a) Blocks of 7x7 pixels. (b) Blocks of 11x11 pixels. (c) Blocks of 13x13 pixels.

For the above element template, we arranged the blocks in order and connected them one by one. They are allowed to overlap or be well separated to meet
certain requirements, as shown in Figure 2.16. In fact, the ideal blocks should be those based on the object’s desirable features. Hence the number of kernel elements is not necessarily determined by the block size. However, the number of kernel elements and the area they cover will affect the tracking performance. Fewer elements will risk less accuracy in locating the object as they may not carry sufficient information to characterize the object. Larger blocks will not only increase the computational cost but also possibly blur the unique feature at the center area. The choice of block size, location and the number of blocks may need to be based on some experimental results prior to an automatic tracking process. They essentially depend on the features of the object of interest.

![Figure 2.16](image) (a) Distribution of the kernel blocks. (b) Corresponding elements

In our experiments with this sequence, since the object stands out well from its background, the object can still be well located even if the block are as large as 15x15 pixels. In Figure 2.17, (a) shows the object in the first frame and (b) shows the object derived for the 85th frame without any boundary refinement, and some of the object centers in the sequence are traced out in (c) in cyan dots on the back of the object.

The block size used here is 9x9 pixels and the blocks are evenly distributed
over the object. The distribution of elements is adjustable based on the features of the object. In this sequence, for instance, the element locations on the neck part of the object could be adjusted with a larger distance to reflect its larger deformation for the subsequent frames since its head moves up and down more quickly and in larger steps than the other parts. Also in the searching step, a weight factor can be imposed on the corresponding kernel elements to reduce its effect on the matching.

![Figure 2.17](image)

Figure 2.17 (a) 1st frame. (b) 85th frame. (c) Trace of object centers in the sequence.

This method is better able to handle the local changes by adjusting the weight factor or adjusting the corresponding element’s position on the specified area.
such as the head of this object, while the first method would be better at handling the global changes like illumination change in the environment.

Figure 2.18 Movement of object center

Figure 2.19 Trace of object centers
In Figure 2.18, we illustrate a deforming object captured in some selected frames of another video sequences by highlighting the object center with a cross. The block size is 7x7 pixels, and the slight center deviation through the kernel template is updated once every few frames. The object center is also traced out in Figure 2.19.

2.6.3 Experiments on kernel-based pixel model

We continue our experiments on the video sequence in Figure 2.10 (a). The object density derived from (2.10) is illustrated with red solid line in Figure 2.20, while the background density in local background is in blue dotted line. The kernel function $\phi(\cdot)$ in equations (2.10) and (2.11) for pixel likelihood use Gaussian function. The object is approximately described by two ellipses, one for the body and a small one for the tail. Then the standard variation for spatial model takes the value of diameter to ellipse border. Here we will vary the standard deviation for different directions. The posterior probabilities, based on the object and background intensity likelihoods with the assumption that the prior probability is the same for both the object and the background pixels, are illustrated in Figure 2.21.

In Figure 2.22, we applied different spatial models on the posterior probabilities. As the object center has been located by maximizing the sum of probabilities via (2.15), it is made use of in Figure 2.22 for the illustrative purpose. In Figure 2.22 (a) the probabilities are calculated by the product of the posterior probabilities and the spatial probabilities obtained from a Gaussian function with one standard deviation in a single basic shape. In Figure 2.22 (b)
the standard deviation for the Gaussian functions is varied for the spatial model as in (2.13). Figure 2.22 (c) illustrates the case of two emulated basic shapes (one for the body and one for the tail) with varying standard deviation for the Gaussian functions, which shows a better result than the previous two.

![Figure 2.20](image)

**Figure 2.20** Object and background intensity density.

![Figure 2.21](image)

**Figure 2.21** Posterior probability.

In Figure 2.23 (a), the object location is obtained by (2.15), where the object is modeled with two ellipses for the body and the tail respectively. The object location in Figure 2.23 (b) is obtained by (2.16), where the object is modeled with one ellipse only for body, but the probabilities involved for outlining the object are constrained by the previous object shape. Although both can reasonably locate the object in the current frame, the more accurate spatial model, as in the first case, would work better.
Figure 2.22  (a) Probabilities with spatial model (single Gaussian). (b) Probability with varying standard variation for Gaussian function. (c) Probability with varying Gaussian and two simulated basic shapes.

Figure 2.23  (a) Locate object with better spatial model. (b) Locate object with constrained probabilities.

Finally we illustrate the object motion every five frames from the 10\textsuperscript{th} frame to the 55\textsuperscript{th} frame in the Figure 2.24 (a) with the yellow points there indicating the center movement of object. They are also illustrated in Figure 2.24 (b) with unrefined contours during this movement.

The method is able to find the correct location of object in the subsequence frames, as long as the spatial model can properly reflect the object shape. The shape at the first frame needs to be identified and be modeled, and it will be adjusted according to the detected deformation of the object at the current frame.
We note that the above three approaches made use of the kernel density modeling in different manners to solve the tracking problem. They can be properly selected to suit different scenarios. For example, the first one can be used when the object can be properly described by basic shapes and by the shape deformation during motion within certain bounds. The limitation of the first approach is that an unsuitably selected shape for the object may lead to a poor performance. Though the annuli are appropriate for most objects, in some situations such as having a snake object, this approach becomes unsuitable. In the second approach, if the deformable part of the object is large or the deformable direction is unpredictable, the tracking accuracy will decrease. In the third
approach, the component of the pixels spatial model has to be adjusted with the shape update during the object movement. Though the kernel method greatly reduces the computational cost in the searching step for the object location than the normal template match because it uses much less samples to represent the object, it has also needs an additional step to derive the kernel elements. The computational cost is related to the size and moving speed of the object. Hence the main advantage of this method is its ability to accommodate certain illumination variation, local appearance change and shape deformation.

2.7. Summary

We proposed in this chapter several kernel-based models for object tracking in video sequence. The problem of locating a particular object in the subsequent frames is approached in different ways for which the corresponding kernel-based models are developed. Each of these models has been designed to achieve a better performance for a certain type of problems with its advanced feature modeling.

In the first approach, an object representation is established by a kernel-based color probability density function over a collection of concentric annuli at the object centre. Each value of the density function corresponds to the pixels in a specific annulus as a single entity. Although the object representation through features on annuli can be extended to more than one patches of stable color appearance, the rotational invariance of the annuli simplifies both the representation and the computation. A more noise-tolerant similarity metric has
been shown to perform more robustly compared with the traditional Euclidean distance, when determining the similarity between two object representations. The proposed tracking scheme has been shown to be applicable to video sequences containing the motion of a non-rigid object within a non-stationary background. The method is both effective and efficient, due to the associated succinct modeling, in dealing with objects of deforming shapes, and the last verification step further refines the shape of the tracked object in the newer frames.

In the second approach, the adaptive kernel template is proposed, where the object is modeled by statistical features or elements spreading over the object. This relatively sparser representation allows the matching process to deal with fewer corresponding elements and thus speed up the object extraction process. It solves the deformation problem in both the interior and the outer region of the object that the traditional method of template match found difficult to handle. It is more flexible to model the stability of the object appearance in comparison with the first approach, and to handle the local appearance changes.

In the third approach, the object and the background in a scene are both modeled with the kernel density estimation. These likelihoods of object and background are used to calculate the pixels’ posterior probabilities belonging to the object within a Bayesian framework. The spatial probability model for potential object is established and utilized to extract the object pixels for the current frame. The object extraction is thus achieved by maximizing the sum of such probabilities for the pixels.
3.

BAYESIAN PROPAGATION WITH DISCRIMINATIVE OBJECT DENSITY

3.1. Introduction

An object may be characterized by a number of features such as color, shape, gradient and other artificially contrived properties. Depending on the particular individual object-tracking circumstances, some features may be more suitable than the others. Tracking an object typically resorts to the features such as colors, edges and significant points, and the significant features such as uniform color regions, regular texture areas, or strong edges are commonly considered good for such purposes. On the other hand, it is naturally anticipated that for effective tracking an object feature that distinguishes itself more from its background is a better choice than those similar to the background. The object features that are similar to the background would decrease the accuracy of capturing an object,
and could even lead to a tracking failure, whether or not the features are significant on their own. In a dull environment, for example, a bright color of the object is a good choice for tracking, while in a bright environment the dark color of the object is obviously more suitable. In this connection, a segmentation method (Zhang and Desai 2001) was proposed to extract bright targets filtered out by wavelet techniques and a Bayes classifier, where an adaptive multi-scale threshold selection criterion is developed for analyzing the image probability density function (PDF). Collins et al (2005) also presented an online feature selection mechanism to evaluate the features used to improve the tracking performance. Based on a set of seed features, they compute the log likelihood ratios of class conditional sample densities from object and background to form a new set of candidate features tailored to the differentiation task of the object from the background. The feature evaluation mechanism is then embedded in a mean-shift tracking system.

In this chapter, we propose to select discriminative features for tracking based on the contrast of probability density between the object and its local background in different color spaces. We choose to use the densities obtained within local regions because such densities are more stable across the frames and can better characterize the object and its nearby background. For instance, if the object is small relative to the full frame, the full background may undergo large changes and lead to very different densities, which would be undesirable for our density propagation framework. The framework is based on the assumption that the object density would not have a large change between consecutive frames, and
this is also assumed for most existing tracking algorithms. Our main strategy here is to make use of optimal segments of density distribution from different sources, and apply them in the framework of Bayesian density propagation. For instance, in a color-based matching approach, if the object consists of a few color sectors that are well distinguished from its local background, that is, the object color density is well separated from its background color density, then these sectors will lead to a better tracking performance by methods such as template matching or kernel density estimator. This feature of color density discrimination may exist in one or more colors or color-spaces. As long as the object is identifiable, at least one of such discriminated features should exist.

In the following, we investigate in detail what kind of density is an ideal candidate for the tracking process and how to extract the corresponding features. First of all, we know there can be different densities from different color spaces. For instance the RGB space, HSI space and CMY space each illustrate certain properties of object colors. As mentioned above, the background colors may be rich in varieties and may continually vary a lot. Hence it sometimes may not be easy to extract distinguished object color from this variety of unstable features. We thus approach the problem by restricting our consideration to the local background. A small window or other basic shape to include the object will be sufficient for this purpose. Of cause if the object is sufficiently large relative to the background there is no need to consider its local background at all. This local background approach will ensure no performance degradation when the object moves fast in the scene because the estimation is made from previous known
frame which is used to extract the discriminative color densities on the statistical basis. The object location can be traced out by following the higher probabilities of the object on the frame, and the densities selected should reflect the distinguishing properties of the object features relative to its local background.

3.2. Choice of Features

An object may be characterized by a number of features such as color, shape, gradient and other artificially contrived properties. Depending on the particular individual object-tracking circumstances, some features may be more suitable than the others. The discriminative property of these features is extracted from the probability densities of the object and the local background. These densities statistically represent the pixel values and the respective quantities. The intensity values themselves are often not enough to characterize the object.

3.2.1 Differentiating features

Although many features such as strong edges, color uniformity or object texture are known to be applicable to capturing an object, the higher contrast between the object and the background via a differentiating feature will better help track an object. Since the object features that are similar to the background would decrease the accuracy of capturing an object, and could even lead to a tracking failure regardless the features are significant on their own, an object feature that distinguishes itself more from the background is a better choice than those more similar to the background. We here propose to select color features that offer better contrast of probability density between the object and its local
background. We in fact will make use of optimal segments of density distribution from different sources.

In order to extract the local features, we first select a simple shape to limit the local region. We suggest that the shape should be based on the characteristics of the object so that it better and uniformly surrounds the object. The object can use the previous estimated object area or be represented by a similar shape which best represents the object shape. In the example frame shown in the Figure 3.1 (a), an ellipse (green ellipse) is used to approximate the local region. Then the object is approximately represented by the small ellipse (red ellipse), and the local background is approximately represented by the annular area between two ellipses. Indeed, this approximation not only eliminates the previous accumulated inaccuracy in the object border estimation, but also simplifies the process as accurate border estimation may not be necessary for the tracking. The shape, however, may need to be adjusted with the updated object for the next frame.

The object density and the local background density can be calculated from the ellipses in the previous frame. As illustrated in Figure 3.1 (b), the solid (blue) line is the object density \( p_o(I) \) obtained from the smaller ellipse, and the dotted (red) line is the local background density \( p_B(I) \) obtained from the annular area, while the dashed (green) line represents the mixture density \( p_{mix}(I) \). Here subscript \( O \) denotes the object and \( B \) denotes the background. \( I \) is the intensity or other parameter from other color space. In general, given two densities \( p_o(I) \) and \( p_B(I) \), the mixture density is defined by \( p_{mix}(I) = \alpha p_o(I) + (1-\alpha)p_B(I) \), where \( \alpha \) is the weight coefficient. What we are interested in is the discriminative feature of the
object density from the background which will be used to obtain the pixels belonging to the object with high probabilities in the subsequent frame in the later steps. The reasons to use density rather than the pure color itself are: i) A color is often not discriminative enough by itself if it exists in both the object and the background; ii) The density indirectly reflects the importance of a certain color in the appearance; iii) It supports the ignorance of the interference of background colors of small quantities.

![Figure 3.1](image)

**Figure 3.1** (a) Local region. (b) Object density, background density and mixture density. (c) Difference of densities $\Delta(I) = p_o(I) - p_b(I)$. (d) The ratio of densities $p_o(I)/p_b(I)$

In order to locate differentiating features, we first define the difference of the two densities by
as shown in Figure 3.1(c). In this framework, the density suitable for object tracking is the one having larger $\Delta(I)$. More precisely, only the density segments with sufficiently large difference will be used to characterize the object. The small $\Delta(I)$ does not have enough discriminating power and would be ignored. When $p_o(I)$ and $p_B(I)$ are both very small, then this density is not suitable for detecting the object presence because their differentiation is too small to make great sense. In other words, there are very few pixels in this color. In fact, the other difference measurement can be adopted too, e.g. the ratio of object density against the background density, $p_o(I)/p_B(I)$. In this case, the densities having larger ratio are a good choice. If we consider the center of the area under $\Delta(I)$ within a range of $I$, then a higher center position implies a larger feature difference between the object and the local background and therefore leads to better tracking performance. If the given densities can be approximated by a Gaussian distribution, then the model parameters may be made use of to determine the suitable density segments for the feature extraction.

3.2.2 Desirable feature densities and their potential complements

The feature densities of the object from different feature spaces would exhibit different characteristics. We can obviously assume that they would not be the same as a color based tracking scheme will never be used to track objects that have exactly the same color as the background. How do we choose a feature so as to best distinguish the object from its local background? We need to find a
candidate density that maximizes the difference between the object and its local background. A density may be derived from the RGB space, HSV space or from other properties such as the smoothness. The total difference of object density from local background in a positive portion can be calculated by $\Omega_+ = \int \max(\Delta(I), 0) dI$, and that in a negative portion is $\Omega_- = \int \min(\Delta(I), 0) dI$. The overlapping crossover part of object density and background density is denoted as $\Omega_x = \int \min(p_o(I), p_b(I))$. Among several different feature spaces, if the ratio of $\Omega_+$ and $\Omega_-$ is closer to the ratio of area of the object and its local background, and $\Omega_x$ is smaller in the range of positive $\Delta(I)$, such densities are more suitable for tracking. If the centers of the areas of the positive $\Delta(I)$ and negative positive $\Delta(I)$ are larger the density is a more suitable feature for tracking. In summary the tracking performance via the densities depends on the ratio of positive and negative difference in area and area centre, and the degree of overlapping. For example, Figure 3.2 shows that the densities of object and background may still be able to separate in saturation, or separate better than the intensity within the HSV space, even though they may appear overlapping too much in the RGB space.

A single pair of densities of object and local background may at times not be sufficient to obtain a good result. We observe in Figure 3.3 (a), that the leftmost portion of the object density for the intensity is quite well separated from the local background density. However, the remaining portion on the right may lead to weaker object detection or even missing areas in the tracked object. To utilize fully portions of discriminating densities coming from different perspectives, we
propose to combine various optimal parts of different densities from different feature spaces. In other words, we enhance the object representation of the differentiation from the background through the use of complementary features.

\[\text{(a) Hue difference. (b) Saturation difference. (c) Intensity difference.}\]

\[\text{(a) Object and background intensity density. (b) Densities difference.}\]
\[\text{(c) Smoothness densities}\]

A predominant reason for investigating multiple feature densities is that a selected object density could overlap with the background density on a small part when the difference between them is not sufficiently large there and this may
lead to certain areas being missed out in the object representation. We note that densities of any feature exhibiting larger difference can be selected, and more such densities used may increase the accuracy. The possible drawback for using too many features is that it may also increase the computation unnecessarily if less is enough for the purpose. As a result, the size of the density overlap and the extent of the difference will determine if there is a need to make use of an additional feature property. The density distributions illustrated in Figure 3.3 (b) and (c) exemplify such a case where the object can be better represented by the combination of these two set of densities.

### 3.3. Bayesian Propagation of Densities

We recall that object tracking is largely based on the features the object possesses. We thus propose to use discriminative local densities of the object and its surrounding background for object tracking. This approach is based on the proximate density induced by the corresponding Bayesian rule. To improve the estimation of the object propagation, other information such as the motion may also be considered.

#### 3.3.1 Density update via the Bayesian rule

We first need to establish a Bayesian framework for our tracking purpose. Suppose there are a set of $n+1$ mutually exclusive hypotheses $H_i$ which form the complete hypothesis space. Then the Bayesian inference aims to update or to newly infer the probability of a hypothesis based on the latest evidence or observation. Let $p(H_0)$ be the estimated prior probability of a hypothesis $H_0$, then
the Bayesian inference rule allows one to improve the estimated prior probability into the posterior probability $p(H_0|D)$ if a set of additionally observed data $D$ is available for the improvement of the probability estimation. Let $p(D|H_i)$ be the likelihood of the hypothesis $H_i$ under the observed data $D$ and $p(H_i)$ be the prior probability of the hypothesis $H_i$. Using a statistically well established formula the posterior probability $p(H_0|D)$ of the hypothesis $H_0$ can be calculated through the use of the prior probability of the hypothesis and the probabilities of the observed data under the different hypotheses. The formula reads $p(H_0|D)p(D) = p(D|H_0)p(H_0)$, which can be further expanded into

$$p(H_0 | D) = \frac{p(D | H_0)p(H_0)}{\sum_{i \neq 0} p(D | H_i)p(H_i)}.$$

For our specific problems in this work, we denote by $O$ the hypothesis that the current pixel of value $I$ belongs to the object, and by $B$ the hypothesis that the current pixel $I$ belongs to the background. Thus $p(R^o)$ and $p(R^B)$ are the prior probabilities of the pixel belonging to the object and the background respectively, that is, the probabilities estimated prior to inspecting the actual pixel value in the current frame. According to the Bayesian rule, the pixel probability belonging to the object at the current frame, $p(R^o|I)$, can now be expressed as

$$p'(R^o | I) = \frac{p'(I | R^o)p'(R^o)}{p'(I | R^o)p'(R^o) + p'(I | R^B)p'(R^B)} \quad (3.1)$$

where $p(l|O)$ is the likelihood of the current pixel of value $I$ belonging to the object and $p(l|B)$ is the likelihood of the pixel belonging to the background. The
denominator in (3.1) is essentially a normalization factor. Although we can not
determine in general the exact prior probability of a pixel being on the object or
the background, we can nonetheless assume they are constant. We rewrite (3.1)
as \[ p^o (R^o \mid I) = \frac{p^o (I \mid R^o)}{[p^o (I \mid R^o) + \mu]}, \]
where \( \mu = p^o(R^B) / p^o(R^o) \) is approximated by the ratio of the number of pixels in the background and in the
object. Since the densities, especially the object density, would be very close to
that known for the previous frame at time \( t-1 \), we can use \( p^{o-1}(x \mid O) \) in place of
\( p^o(x \mid O) \) for the practical calculation. The same also applies to \( p^o(x \mid B) \). We note
here that this method is advantageous over direct density matching because less
iterative searching would be required. When the local region excludes most
background by its shape border like an oval, the local object density will in
general fall into this category.

If multiple feature densities are used to achieve a better representation of the
object the probability of pixels belonging to the object is the sum of probabilities
calculated from all selected features. The multiple densities used can be viewed
as the complementary features that are extracted individually and independently.
The sum of probabilities are expressed by

\[
p(R^o \mid I) = \sum_{i=1}^{N} \lambda_i p_i (R^o \mid I) \Theta(\Delta(I)),
\]

where \( N \) is the number of features, \( \lambda_i \geq 0 \) are the weight factors, \( \sum_i \lambda_i = 1 \), \( \Theta(\cdot) \)
represents the selected density range, \( \Theta(z)=1 \) for \( z > 0 \) and \( =0 \) otherwise. This
would, in general, result in a more complete coverage of the object. The resulting
probability sum may not be the density that integrates to one but this will not
affect its essential usage. For a typical rough tracking problem, one distinguishing feature often suffices.

### 3.3.2 Motion features

We note that the accuracy of the above method may suffer if the object density and local background density are less well separated. For instance, if the object is very similar to its local background with regard to the selected feature, the above method alone may not be sufficient to extract the object. We thus need to resort to using additional information such as the pixel space relationship as well as the motion information. The motion within a static background is typically estimated from the data difference of successive frames, i.e., \( \Delta(s, t) = I(s, t) - I(s, t-1) \) at the pixel location \( s \). Then the inter-frame data difference is modeled as a mixture density of the static component density \( p_M(d) \) and the motion density \( p_M(d) \), where \( M \) and \( \bar{M} \) denote the motion and static components respectively, and \( d \) represents the difference value in intensity. The model parameters can then be obtained through the use of maximum likelihood method by maximizing the joint density. The Expectation Maximization algorithm is often effective in solving such mixture density, and a similar use can be found in (Paragios and Deriche 2000). To integrate the motion information into the refinement of the object density, we calculate the object pixel probability \( p'(R^o|I) \) for the time step \( t \) by

\[
p'(R^o|I) = \frac{p(R^o|I)p(d|M)p(M)}{p(d|M)p(M) + p(d|\bar{M})p(\bar{M})},
\]

where the priors \( p(M) \) and \( p(\bar{M}) \) are assumed constant, and \( p(d|M) \) and \( p(d|\bar{M}) \)
are calculated from Gaussian distribution of density with the model parameters. We finally note that other methods may be required to extract the motion information from a video sequence of a moving background. The object probability obtained here, however, would distinguish the object better from the local background than those without the use of the additional motion information.

3.4. Region-wise update

Our Bayesian framework will lead to a probability image that indicates the pixel probability for the desired object. The region an object occupies is obtainable by thresholding the object pixel probabilities while ensuring the coherence of its neighbors with such as region growing, and region consolidation method (Huang and Jiang 2005). Since sometimes part of object area may be very close to the background and may not be well represented by the probability distribution, the probabilities may not always represent the full object shape. To overcome such a problem, we simply project the object oval in the previous frame onto the current region, and adapt the oval position along the object movement throughout the sequence. To fit an oval onto an object we just need to ensure that the oval differs with the object region as least as possible in terms of the covered area difference. Oval is a natural choice for a basic shape to include the object, although other basic shapes may play the same role as well.

It is, in fact, an iterative process to adjust the oval from a previous frame so as to well represent the current region by a preset ratio of the number of pixels of region out of the oval and the number of pixels of background included inside the oval. The oval parameters needed to optimize include its centre position, rotation
angle, major axis and minor axis. In total there would be five dimensions for the problem. In fact in the case of a regular shape such as an oval, there is a simplified and faster way to locate the oval on the current region obtained. We first calculate the centre \((x_0, y_0)\) of the current region obtained by (3.1) or (3.2), which is considered as the centre position of the ellipse,

\[
x_0 = \frac{1}{n} \sum_{i=a}^{b} x_i, \quad y_0 = \frac{1}{n} \sum_{i=a}^{b} y(x_i),
\]

where \(n\) is the number of pixels in the object region, and \((x_i, y(x_i))\) represents the coordinates. Then we divide the region through this centre into two parts with a line having the same direction of the minor axis \(b\) of the ellipse from the previous frame. Then we calculate the centers of these two parts separately. Next we use the slope of the line determined by these two centers as the new rotation angle \(\theta\) of the ellipse. Now what we need to refine further is to adjust the major axis \(a\) and the minor axis \(b\) of the ellipse to make it better describe the region, which actually can be done separately and can achieve required accuracy by the proper preset threshold \(\tau\). In implementation, the object region pixels outside the ellipse and background pixels inside the ellipse in two directions of diameters can be marked by a selected rectangle separately as shown in Figure 3.4. Their ratio \(\omega\) is then easily obtained. The axis is adjusted so that \(\omega \rightarrow \tau\). Comparing this method to the more general method, we observe that it has greatly reduced the computational process and improved the speed.
3.5. Implementation and analysis

We will now devise a number of experiments to examine the framework of discriminative density propagation for object tracking. These experiments will also illustrate the effect of a chosen feature on the tracking performance. The factors for consideration here include the density selection from different sources, the complementary densities, as well as the less-discriminating densities in terms of the additional motion information. The video sequences for the tests are illustrated in Figure 3.5 and Figure 3.1 (a).

Choice of density segments

We first illustrate the impact of density segments chosen for the object representation. The object density and the local background density for hue, saturation and intensity for Figure 3.5 (a) are shown respectively as in Figure 3.6 (a), (b) and (c), and their corresponding difference curves are shown in Figure 3.6 (d), (e) and (f). The ratio of the amount of pixels between positive and negative density from the saturation space as in Figure 3.6 (e) is closer to the ratio of the amount of pixels between the object and local background than those from the hue (d) and the intensity (f). Also the density overlap in the saturation is smaller. This shows that the saturation density can exhibit better discrimination for the
Figure 3.5 Test video sequences. (a) Penguin. (b) Meerkat. (c) Bird.
(d) Car.

Figure 3.6 (a) Hue density. (b) Saturation density. (c) Intensity density. (d) Hue difference. (e) Saturation difference. (f) Intensity difference. (g), (h), (i) Object probability from hue, saturation, intensity respectively.
object from the background than the hue and intensity densities. The resulting probabilities of object pixels from the Bayesian framework are shown in Figure 3.6 (g), (h) and (i). These intuitive results indicate that the feature in the saturation space is the best for object tracking in this framework and can be used alone to extract the object from subsequent frames. While the densities derived from Figure 3.1 (a) show different characteristics as those illustrated in Figure 3.7 (a), (b) and (c), the hue density obviously better discriminates the object against the background than the other two densities, and Figure 3.7 (d), (e) (f) indeed show that hue is more suitable for the tracking in this case. We note that the video frames in Figure 3.1 (a) and Figure 3.5 (a) look quite similar, but the choice of feature is different.

![Figure 3.1](image1.png)  ![Figure 3.2](image2.png)  ![Figure 3.3](image3.png)

Figure 3.7 (a) Hue density. (b) Saturation density. (c) Intensity density. (d), (e), (f) Object probability from hue, saturation and intensity respectively.

In Figure 3.8, the densities of hue, saturation and intensity were calculated from the video in Figure 3.5 (b), and the corresponding pixel probabilities for the object, obtained via (3.1), are shown in Figure 3.8 (d), (e) and (f). Although the
saturation would give rise to the most discriminative among the three features, it derives only a part of the object rather than the whole. It is thus worth finding a complementary density for a better representation of the object. The combination of hue and saturation features hence yields a good result for object representation as shown in Figure 3.8 (g).

![Graphs showing density distributions](image)

**Figure 3.8** (a) Hue density. (b) Saturation density. (c) Intensity density. (d) Object probability by hue. (e) Object probability by Saturation. (f) Object probability by Intensity. (g) Object probability by hue and saturation. (h) Object probability by hue and intensity. (i) Object probability by saturation and intensity.

**Complementary density segments**

Our next example below illustrates the use of complementary density segments. The pixel probability for the object in Figure 3.9 (b) obtained by the
intensity density in Figure 3.9 (a) is not sufficient for proper object extraction. When we combined the uniformity feature, e.g. the local standard variations, of intensity as in Figure 3.9 (c), the resulting pixel probabilities for the object as in Figure 3.9 (e) can represent the object much better than the one in Figure 3.9 (b).

Figure 3.9 (a) Intensity densities from object and background. (b) Object pixel probability from intensity. (c) Smoothness density. (d) Object probability from smoothness. (e) Object pixel probability from intensity and smoothness.

Integrated motion information

When discriminative features are not present, then other information such as the object motion needs to be available in order that the framework is still applicable. For instance, in a video of static background, the motion can be estimated from the inter-frame difference data. As mentioned before, we model the difference data as a mixture density of the static and motion densities, both are Gaussian, then apply the Expectation-Maximization estimation method to obtain the model parameters. The obtained static and motion densities are
illustrated in Figure 3.10 (c) for the video frame in Figure 3.5 (b). Though the calculated pixel probabilities for the object from intensity density in Figure 3.10 (a) are not directly usable yet in Figure 3.10 (b), the resulting probabilities in Figure 3.10 (d) by (3. 3) with additional motion information are obviously better at highlighting the object to be tracked.

As the final stage, we now illustrate object capturing and the approximated shape adjustment with the additional frame information throughout the video sequences. The probabilities in Figure 3.11 (a) for the video sequence in Figure 3.11 (b) are obtained by threshold, and the approximated shape is adjusted automatically using the proposed method to reflect the newly detected area. The obtained shape borders are projected onto the video frames in Figure 3.11 (b) to highlight the tracking effect. Another similar example is shown in Figure 3.12.
Figure 3.11 Penguin sequence 1
Figure 3.11 (b) and Figure 3.12, the red grids are used to indicate the capturing area and the white ellipses represent the approximated object shape. We note that the discriminative density features turned out to be different for these two penguin sequences.

For a different type of object as in the video in Figure 3.5 (d), we used a rectangle to approximate the car shape; the tracking in the moving background is
shown in Figure 3.13. The size of the rectangle was adjusted with the movement of the car.

For visual clarity, we finally illustrate the moving trajectory of the ellipses in a frame in Figure 3.14 for some of the above experimented sequences. They show clearly the moving state of the corresponding objects.

![Figure 3.14 Traces of the moving objects.](image)

3.6. Summary

We proposed to make use of the discriminative portions of the color densities of the object against its local background to track the object within the probability of multiple hypotheses. The object is approximated by a simple shape, and the desirable portions of color densities are properly selected from different color spaces in the previous frame based on certain criteria. The probability of a pixel being on the object in the current frame is then calculated with the Bayesian estimation where the likelihood of the object pixels and the background pixels are approximated by the corresponding densities. The color densities of the object and the background should be updated with the newly detected area for the estimation in the next frame. For the object of non-discriminative features in the color spaces, additional available information for
such as the object motion can be utilized to facilitate this tracking process. The experiments demonstrated the efficiency of the method due to its simpler modeling as well as its relatively lighter computation due to the absence of the iterative searching. However the tracking performance in this method may be poor if it is difficult to find the proper discriminative densities between the object and the local background.
4.

DYNAMIC DOMINANT FEATURE ELEMENTS

4.1. Introduction

In chapter 2, we have investigated the kernel-based modeling of object appearance for tracking from different aspects. The adaptive kernel template there relaxed the strict pixel-wise comparison from the traditional template approach and the object location in the current frame is obtained by an iterative searching process for the minimum difference of the established model and the candidate object. Although different search strategy may be possibly adopted based on the particular scene to speed this process, in many cases the search may have to cover the full frame. We here in this chapter will further investigate an improved matching process that requires no iterative searching. The features selected for this process are extracted in terms of the crucial elements that
characterize the object in the tracking environment. The main purpose of this work is to approach the tracking problem not only with fewer limitations as those in the kernel-based approach, but also with less computational cost. We will further extend the concept of discriminative features introduced in chapter 3, and will operate on a simpler yet sufficiently discriminative object representation on the local background within a statistical framework. The simplified object model will result in lower computational load and allows for non-rigid movement of an object in a moving background. The probabilistic similarity of object pixels will be directly derived from the corresponding crucial features, and the object tracking is thus carried out based on the dynamic segmentation of the object with respect to its local background and in terms of the dominant colors. The progression flow of the matching process is summarized in Figure 4.1. The work done here has been published in (Huang and Jiang 2007).

**Figure 4.1** Progression flow of the matching
This chapter is organized as follows. In section 4.2 we propose a new form of representation for the target template within the environment of a local background, and analyze the extraction of the color intensity bands for the object. In section 4.3 we then develop a matching scheme in a probabilistic framework, based on the dominant elements of intensity bands, and follow it up with a refining finalization. Finally in section 4.4, the implementation results and efficiency will be shown.

4.2. Dynamic dominant elements

Among the object features such as color, edge and shape, the color feature is often more significant in matching the object. Many existing algorithms perform matching based on pixel to pixel, block to block, or region to region correspondence. We here propose to model an object through a simplified representation of the significant color bands in terms of their significance, and in doing so also take into account the local background so as to conduct the matching process in an efficient and practical way. Our proposed method will segment the object colors into certain suitably chosen intensity bands, determined by maximizing the band difference between the object and local background in the previous frame. Then the dominant elements are derived from the distinctive bands and used for the tracking. The detailed steps are described below.

4.2.1 Band selection

The segmentation bands are selected based on the distribution of the object colors. In order to improve the matching accuracy in the process, we take into
account the neighbourhood of each individual pixel, as the color distribution can be calculated and replaced by the local mean value of the pixel with its close neighbouring pixels. Next we find the color intensity $I_s$ which peaks the color intensity distribution, and use it as the centre of the most significant band. This enables us to extract the largest area of similar colors on the object which we will refer as a dominant element.

We initialize the intensity band width to $w$, which can be either fixed or varied with the color distribution. We can then define the bands $H=\{h_1, \ldots, h_n\}$ due to the partition on the intensity. More precisely, for a fixed $w$ the bands $h$'s are defined for $i=1, \ldots, n$ as

$$h_i = [I_s + (2i - 2k_b^i - 1)w/2, I_s + (2i - 2k_b^i + 1)w/2],$$

where $n = k_b + k_e$, $k_b = \lfloor (I_s - w/2)/w \rfloor + 2$, $k_e = \lfloor (1 - I_s - w/2)/w \rfloor + 1$, $h_1(1,1)=0$, and $h_n(1,2)=1$. If the RGB color space is used, then each color component will be dealt with independently. We can rank the $\{h_i\}$ in the descending order of the number of pixels contained in the band to rate the contribution of the color to the object. Our matching process aims to identify the same object from the subsequent frame under the regularity assumption that the appearance of the object and the local background would not change drastically in different ways. Our main strategy for the band selection is to take into account the local background so that we can represent the object with the bands that most distinguish it from its local background. To simplify this task, we also use a simple shape, such as a rectangle, to demarcate the object along with its nearby
local background. Before we rank the \( \{ h_i \} \), we first describe below the choice of a proper band width \( w \).

With a selected \( w \), we can obtain \( H \) by (4.1). Then we will make use of \( H \) to segment the object and the local background. To start with, we could just trivially assign an index to the segmented region in the natural order of the bands. In order to achieve a better segmentation so that the object segments will better distinguish themselves from those for the local background, we need to incorporate the pixel spatial information so as to choose better bands for the segmentation, as the color distribution of the object and the local background may not be enough for this purpose. Figure 4.2 (a) shows a color segmentation in a local area. The technical method we introduce here is to calculate the ‘difference’ of the neighbouring regions, one on the object side and one on the background side, along the object boundary. To carry this out, a set of boundary pixels \( z_{i'} \), \( 1 \leq i' < n' \), \( n' \) is the number of boundary pixels, typically successively equal in distance, along the boundary of the object are first selected, and a circle or a square is used to trap a local neighbourhood where, for instance, the circle centers at a selected boundary pixel \( z \) and the radius \( r \) is half the distance of the successively selected boundary pixels, as illustrated in Figure 4.2 (b). For notational simplicity, such squares of width \( 2r \) will also be referred to as a ‘circle’ of radius \( r \). If we denote the segment membership as a bit pattern \( \theta \), with each single bit indicating the presence of a particular segment, then we can compare the segment membership \( \theta^+ \) on the object side and the membership \( \theta^- \) on the background side, as shown in Figure 4.2 (c). The task thus becomes searching
an optimal $w$ which leads to the largest difference between $\theta^+$ and $\theta^-$. The involved calculation however does not have to be exhaustive as we only need to approximate $w$ in a rough scale.

![Images](a) A color segmentation. (b) Typical local neighborhoods along the boundary. (c) Segmentation patterns in a sample circle.

Figure 4.2 (a) A color segmentation. (b) Typical local neighborhoods along the boundary. (c) Segmentation patterns in a sample circle.

To simplify the calculation of the segmental difference along a boundary section, we count up such differences (the difference of segment indices, or the difference of color) while ignoring their magnitude. Since $\theta$ is a bit pattern indicating which color segments are contained in the neighbourhoods of the boundary, the number of differing segments will obviously increase as the bandwidth decreases, as shown in Figure 4.7. We hence apply the total number of segments in the neighbourhood part of the object as a normalizing factor for the difference comparison. Since the number segments in the background are not important here, they will not contribute to the normalizing factor. Hence our calculation can be expressed mathematically as

$$d_{\theta} = \frac{\sum_{i=1}^{n'} \sum (\theta^+_i \oplus \theta^-_i) / (\sum \theta^+_i)^2}{\sum \theta^+_i},$$

where the *total segment disparity* $d_{\theta}$ indicates the extent to which the object separates itself from the background. $n'$ is the total number of circles selected.
along the boundary path, and $\oplus$ represents the bit XOR operation. We note that the radius $r$ of such circles may have some impact on the calculated disparity, but the trend of the disparity that better separates the object from the background would not change much except when it is cluttered near the boundary. The main factor affecting the disparity is the bandwidth. A smaller intensity band width results in more segments and segment difference, while a larger bandwidth would also smooth the difference between the object and its surrounding background, as shown in Figure 4.7 (f). Therefore the $w$ having the largest $d_\theta$ is treated as the most desirable band width. In fact the band width $w$ may also be selected in a range corresponding to a higher $d_\theta$.

Once such a band width is determined, a set of intensity bands suitable for the object identification can be selected via (4.1). We rank these bands in the descending order that indicates the significance of the colors, i.e. the pixel amount falling into the band in the object and the extent of difference between the object and the background. The actual ranking is done by introducing weight factors to combine the effects due to the number of pixels within a band in the object and to the density differences of the band.

### 4.2.2 Extraction of dominant elements

We now seek to represent an object with a few feature elements that would distinguish the object from the background and facilitate tracking the object. We first consider two ways to extract such elements with the help of intensity bands obtained earlier. The first is to use the centre of each band, or the point at which the color density of the object maximises within the band as the feature element.
The object colors and the local background colors are binned based on the corresponding bands. We now define by $\Delta_k$ the difference of the number of pixels in the $k$th band, $\Delta_k = Q'_k - Q^b_k$, where $Q'_k$ and $Q^b_k$ are respectively the number of object pixels and the number of local background pixels that fall into the $k$th bin. We also define $\delta_k$ as the ratio of the number of pixels in the $k$th band, i.e. $\delta_k = Q'_k / Q^B_k$, where $Q^B_k$ is the number of pixels in the background of the full frame that fall into the $k$th bin.

To formulate the criteria for the band selection, we first set the threshold $\mu$ for all $\Delta_i$ and set the threshold $\nu > 0$ for all $\delta_i$ respectively. The $k$th band will be selected, and thus considered dominant, if both $\Delta_k > \mu$ and $\delta_k > \nu$ for all $k$. Both conditions together ensure the $k$th band to highlight the object from background. If the matching process in the later steps (see Figure 4.4) is carried out only on the local background, the second criterion can be simply ignored. But if the object moves by a far distance and is thus located outside the local background, a larger window or the full frame has to be considered by the second criterion. For the selected $m$ bands $\{h'_k\}$ with $m < n$, we denote by $l'_k$ the centre of the corresponding band $h'_k$. Hence these $l'_k$ for $k = 1, \ldots, m$ will serve as dominant elements to represent the object. We recall that these $l'_k$ can also be selected to correspond to the highest density in the respective bands $h'_k$, as this allows $l'_k$ to represent the dominant component in the segment.

We note that the bands obtained above are selected according to the intensity distribution without considering the actual spatial information. Our second way is therefore to extract dominant elements to build the object model by integrating
with the spatial information of the pixels. This is carried out after we obtain \( \{ h'_k \} \) and the selected segments inside the object. We choose a small window for every pixel within a selected segment, keep within the window only the pixels in the same segment, and then calculate there the local mean, or another feature value by a different model such as a Gaussian, to substitute the pixel with the calculated value. Amongst such new pixel values, the closest one to the middle of the band is chosen as the element representing the segment. When the matching process is carried out in the next step we would need to pre-process the pixels in the new frame in the same way, i.e. with the same local window and with the same pixel modeling, e.g. choosing the local mean as the true pixel value. The difference of this method to the first one is that it takes the neighbor pixels into account for the value of the elements, while in the first method the elements are directly derived from the selected bands in terms of pixel intensity. The advantage of the second method is that it is able to reduce the effect of the color noises in the background when only sporadic background pixels exhibit intensities very similar to those in the object since these pixels are excluded by new value of elements. Indeed, the second method integrates a second feature into the element model, which is the pixel context or color smoothness.

Let \( \{ h_k' \} \) denote those in \( \{ h_k \} \) that do not meet the selection criteria for the dominant elements. If the segment corresponding to an \( h_k' \) within the object is located next to the object boundary, we may examine the surrounding segments in the background. If there are no similar segments in the background, which means it is still distinguishing locally, then we can use it as a supplementary
band. Such supplementary bands may be added to the set of dominant bands to provide fuller representation of the object.

4.3. Similarity Rendering

We have introduced in the previous section an efficient model to represent an object so that searching for it in a different frame will be at a lower computational cost. In order to achieve better matching capability we will establish a probabilistic framework that also accommodates well the object representation in terms of dominant feature elements. In the chapter 2, the kernel density estimation has been used in the appearance modeling, and the similarity of the pixel colors of the object and the background to the corresponding known samples can be calculated via the kernel density, as in (2.4) and (2.5). However the similarity here is largely determined by the number of samples in the same color bin and so the similarity is actually represented in terms of density. To realize the color similarity for the pixels in the current frame to the known object elements we will directly render the similarity probabilities on the pixels as detailed below.

4.3.1 Similarity function

Let \{I_i\} denote all the pixels in the current frame or the part of the frame corresponding to the local window in the previous frame. Then the probability of a pixel \(I\) in the current frame to belong to the \(k\)th element of the object in the previous frame is proportional to,
$p_k(I) = \varphi(|I - I_k^c|)$, 

(4.3)

where $\varphi(\cdot)$ is a monotonic decreasing function with $\varphi(\cdot) \rightarrow 1$ when $r \rightarrow 0$ and $\varphi(\cdot) \rightarrow 0$ when $r \rightarrow \infty$, such as a Gaussian or triangle function. Hence $p_k(I)$ measures the similarity of the pixel $I$ in the current frame to the $k$th element of the object obtained in the previous frame, which is referred to as the component probability map. The $k$th element essentially represents a range of intensity on the object in the previous frame. Thus the probabilities in the component probability map reflect the similar pixels in the current frame corresponding to that range of intensity on the object in the previous frame without the need to compare every pixel in the previous frame.

Since the object to be tracked is characterized by its dominant elements in the scene, we are interested in formulating through the use of $p_k(I)$ a probabilistic distribution that measures how similar a pixel in the current frame is to the selected elements in the previous one. For this purpose we sum up the probabilities of its belonging to each such element to reflect its overall probability of being within the object region in the current frame:

$$p(I) = \sum_{j=1}^{3} \sum_{k=1}^{m} \alpha_k^{(j)} P_k^{(j)}(I),$$

(4.4)

where index $j$ corresponds to the three RGB colors, the coefficients $\alpha_k \geq 0$ are chosen to reflect the importance of the object elements derived from the selected bands, $m$ is the total number of the elements, and $\sum_{j,k} \alpha^{(j)}_k = 1$. We here typically adopt the $\alpha_k$ evenly for the selected bands. This sum will be referred to as the
probability map later on. Hence $p(I)$ essentially estimates the probability of the pixel $I$ being similar to the color characteristics of the object. The reason to use the sum of probabilities for colors rather than the product of probabilities is that this element matching process in a color will exclude the non-object pixels by using a narrow band width for the similarity function. Another reason is that the element extraction will be conducted independently in each color space. Thus some intensity ranges may be excluded for the element presentation in a difference color space, which lead to lower probabilities of those pixels within this range. The product of probabilities of colors will result in a small object pixel probability. Since the matching feature can be obtained and trusted from individual colors, the sum is more meaningful.

### 4.3.2 Object extraction

To extract the object region from the probability maps, three techniques can be utilized to establish the candidate region for the object. The main strategy is to treat the non-object part, i.e. the background pixels, as those pixels whose probability of being within the object is uniformly 0 nearby. The first technique uses constraint similarity function. In order to yield a better object probability representation, we reduce the effect of the pixels that may accumulate sufficient probabilities but actually are not really close to any of the dominant elements by limiting the contribution of these pixels to the similarity in (4.3), determined by a distance $\chi$, i.e.

$$
 p_k(I) = \begin{cases} 
 \varphi(|I - I^c_k|) & \text{if } |I - I^c_k| \leq \chi \\
 0 & \text{if } |I - I^c_k| > \chi
\end{cases}
$$

(4.5)
For instance, setting $\chi = w/2$ would exclude the pixels which have a distance to the $k$th element larger than half of the band width. In other words, the probabilities calculated are only for those pixels within the $k$th band. The second technique is to set a single sufficient threshold on all the probabilities $p_k(I)$ for individual element before they are summed up in (4.4). The third technique is similar to the second one, but allows a different threshold for each individual segment and applies a final threshold to the total sum $p(I)$.

As the dominant bands are selected mainly based on the local region in the above approach, it is possible for the probabilities obtained in an area distant to the object to also exhibit certain similarity to some elements. To reduce this interference, we introduce a spatial filter to filter out the distant pixels. In this connection, we first detect the rough center of highest probability region (candidate region) nearest to the previous object location. Next a masking function is applied with respect to that center. This thus results in the following modified probability distribution for a better object extraction.

$$ p'(I) = g(d(I))p(I), \quad (4.6) $$

where $d(I)$ is the distance from $I$ to the estimated center, and $g(\cdot)$ is a monotonically decreasing function.

If the object segments are well distinguished from the local background ideally the obtained probabilities could be able to well represent the object to be extracted from the current frame. Yet the background often does interfere with the object extraction due to having certain amount of similar colors or texture
pattern as that in the object. The band selection process discriminates against those bands that are similar to the background, though the presence of different RGB colors may compensate some of these. Moreover the intensity bands obviously do not carry any spatial information. Hence we will undertake to refine the extracted object with the help of the object shape from the previous frame. More precisely, the former shape is first projected onto the current probability map $p'(I)$ constructed earlier on. Then the position of the shape is adjusted so that the total probability of the current region included by the shape is the maximum. Next we render the probabilities with the neighbors, i.e. the probabilities will adapt to a new value by combining the weighted values of the neighbors. Based on the probability map, a threshold $\tau$ is chosen to identify the candidate object. We can expand its contour when the probability at the contour point is larger than the given threshold $\tau$ and shrink the contour when less than $\tau$. We can thus exclude the nearby scattered background pixels from the candidate object. The choice of the threshold will depend on the similarity function, constrained probability in (4.3) and (4.5), and the spatial filter (4.6). It may lead to ragged boundaries as it does not ensure the smoothness of the contour.

Alternatively, we can propagate the contour in the framework of an energy function. Firstly we expand the shape as the initial contour. Then let mapping $E: [0,1] \rightarrow \mathbb{R}^2$ be a parameterized close curve, the energy function can then be expressed as

$$
\phi(z) = \int_0^1 g(|\nabla p(z)|) |\dot{E}(z)| \, dz,
$$
where $\nabla p(z)$ is the probability change at the pixel on the contour, $g(\cdot)$ is a monotonically decreasing function, and $\dot{E}(z)$ is the vector derivative of the curve with respect to its parameter. The object shape will be obtained by minimizing the energy function. A similar approach in terms of the energy function can be found in (Paragios and Deriche 2000). For better algorithmic stability, we may set a maximally allowed contour shift in a single frame step so as to prevent the contour points from moving an unsuitably large distance in the case of substantial background interference. The main steps to locate in the subsequent frame the candidate target that matches the object in the previous frame are summarized in Figure 4.3 and Figure 4.4.

4.4. Analysis and Implementation

In the following, the approach of dominant elements is implemented and demonstrated for the effectiveness in capturing the object.

Band width

First of all, an analysis of optimal band width $w$ was performed on a video sequence of 240 (rows) x 320 (columns) pixels as shown in Figure 4.5 (a) and (b). The object at the beginning was extracted from the static background as in Figure 4.5 (c). Then the object is on the moving background in the subsequent frames. On the frame within which the object is already known we created a local background by including the object within the 115x135 area. The object intensity density $\rho^o$ and the local background intensity density $\rho^b$ were calculated. We obtain the object density peak $I_o$, as shown in Figure 4.8 (a). The intensity bands
Object modeling

1. Set a local window $W$ to contain the object known in the previous frame.
2. Calculate color distributions, $\rho^o$ and $\rho^b$, for the object/local background respectively.
3. Find $I_s$ at which $\rho^o$ peaks: $I_s = \text{argmax}(\rho^o)$.
4. Loop through different $w$’s:
   a. Partition intensity into bands $h_1, \ldots, h_n$ starting from $I_s$ as one band centre.
   b. Segment local region $W$ with $h$ derived from $w$.
   c. Select a set of circles of radius $r$ along the object border.
   d. Calculate bit patterns $\theta^+$ and $\theta^-$, within each circle, for segments included in the object and background respectively.
   e. Calculate the total disparity of segments:
      \[
      d_0 = \sum_{i=1}^{w} \left( \sum (\theta^+ \oplus \theta^-) / (\sum \theta^+) \right) .
      \]
   f. Set the superior width $w_s$ to $w$ if current $d_0$ is larger, or $w_s$ is not initialised yet.
5. Segment object and background with $h(w_s)$.
6. Calculate $\Delta_k = Q^o_k - Q^b_k$, $\delta_k = Q^o_k / Q^B_k$, $\forall k$.
7. Set parameters $\mu$, $\nu$, and complete band selection $h'_k$, $1 \leq k \leq m$, where $\Delta_k > \mu$, and $\delta_k > \nu$.
8. Obtain centres $\{I_{e_k}\}_{k=1..m}$ for the selected bands $h'$ as dominant elements for the object.
9. Rank $\{I_{e_k}\}_{k=1..m}$ in descending order based on the segment size and the distinguishing power, $\Delta_k$, to the background.

Figure 4.3 Object modeling

$\{h_k\}$ can be calculated by (4.1) with a $w$ and $I_s$ and then a segmentation was performed on the frame. With a circle $r$ the normalized total disparity $d_0$ along the
Matching process

1. Calculate the pixel similarities to dominant elements,
   \[ p_k(I) = \varphi(|I - I^c_k|) \]

2. Sum up the probabilistic similarities, \( p(I) = \sum_{j=1}^{m} \sum_{i=1}^{n} \alpha_k^{(j)} p_k^{(j)}(I) \)

3. Apply threshold on the probabilities or use constrained similarity function. \( p_k(I) = \begin{cases} \varphi(|I - I^c_k|) & \text{if } |I - I^c_k| \leq \chi \\ 0 & \text{if } |I - I^c_k| > \chi \end{cases} \)

4. Apply the spatial filtering if the replacement is able to be estimated. \( p'(I) = g(d(I))p(I) \).

5. Project shape and adapt to new position by maximum probabilities.
   \[ (C_x, C_y) = \arg \max_{y \in C_y} \sum_j p'(I) \]

   where \( \Omega \) is the projected object region, and \( T_{a,b} \) is a spatial translation by \( a \) and \( b \) pixels horizontally and vertically respectively.

6. Choose a threshold \( \tau \), e.g. \( \tau = 0 \), and refine the candidate shape by classifying contour pixels \( y'(z) \) according to \( y'(z) \in \Omega \) iff \( p(y'(z)) > \tau \).

**Figure 4.4** Matching process

Object border is calculated by (4.2). Figure 4.6 illustrates the results of this process with different band width \( w \) and radius \( r \). The curves in Figure 4.6 (a) from top to bottom illustrate the effect of the circle radius changing from two to five pixels. The curve in Figure 4.6 (b) is the lowest in Figure 4.6 (a) plotted in a more illustrative scale. We see from Figure 4.6 (a) that some band widths can achieve larger local differences of the object from the background than the others. We can also observe there that a suitable band width would be approximately 25 to 35 pixels. The segmentation with different band width \( w \) on
the local area is illustrated in Figure 4.7. Indeed, the segmentation is based on local mean values of pixels for better representation. From (a) to (f), the intensity band width \( w \) is from 5 to 55 with step of 10 respectively, and the object contour is drawn for illustration. Intuitively, (d) shows that the object is most separated from its background than the others. (a) is a bit cluttered, whereas in (f) the object and background segments are mixed in some regions.

![Figure 4.5](image)

**Figure 4.5** (a) First frame. (b) Second frame. (c) Object on its own.

![Figure 4.6](image)

**Figure 4.6** (a) Total segmental disparity against band width \( \text{Radius}=2,3,4,5 \). (b) \( \text{Radius}=5 \).

The location of intensity that corresponds to the largest object density is \( I_s=0.12 \), which is depicted in Figure 4.8 (a). For a given band width \( w=30 \), a set of bands were obtained as in Figure 4.8 (a), where dot-dash line denotes the centre of the bands. The corresponding segmentation was then performed on the
frame, with the results shown in Figure 4.8 (b). Figure 4.8 (c) and (d) were drawn for another color – red, with different density peak $I_\nu$. Both colors were done for the band width $w = 30$, which enables a better separation between the object and the local background. The object contours in Figure 4.8 (b) and (d) were drawn there for better illustration and a brighter color there indicates a larger amount of pixels in a segment within the object.

![Figure 4.7 Segmentation with $w=5,15,25,35,45,55$.](image)

**Dominant elements**

With an effective band width 30 derived in the above, all the bands can be identified (see Figure 4.8 (a) and (c)). For each band, we counted the number of pixels within that band for both the object and the local background, i.e. $Q^o_k$ and $Q^b_k$ respectively. For the band selection we set $\mu=0$ as the minimum difference of number of pixels for the object and the local background, if the full frame is considered for the matching, also set $\nu=0.5$ as the minimum ratio of the number of pixels in the object against that of the background of the frame. The centers of
the selected bands were depicted as the long dot-dash lines in Figure 4.8 (a) and (c). These bands are thus used to represent the object so as to differentiate with the local background.

We now use the centers of the selected bands, i.e. dominant elements, to locate

\[ \chi = w/2 \] for (4.5) which are not too far away from the individual elements. The similarity function used here is the triangular function \( \varphi = 1 - |t| \) for \( |t| < 1 \) and =0 otherwise.

Figure 4.9 shows the component probability map by elements from red, green and blue color, respectively. Each map captures a distinguished feature of the object compared to

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{image1}
\caption{(a) Bands chosen for the green color. (b) Segmentation with bands in (a). (c) Bands chosen for the red color. (d) Segmentation with bands in (c).}
\end{figure}
the surrounding background corresponding to the model elements.

Figure 4.9 Probabilities by elements.

Figure 4.10 (a) Total probabilistic from the selected bands. (b) Probabilistic after filtering. (c) Applied a threshold on the candidate region. (d) Projected shape. (e) Shape position after adjustment. (f) The refined object.

The probabilistic sum by (4.4) is shown in Figure 4.10 (a). The distant irrelevant pixels can also be weakened by a spatial filter (4.6) as in Figure 4.10
(b), and a final threshold on this map leads to cleaner Figure 4.10 (c).

We note that the tracking of the object in the next frame can be improved for more accuracy with the proper object shape extracted in the current frame. For this purpose, we first projected the previous shape on the current probability map as in Figure 4.10 (d), and then adjusted the shape location so that the sum of probabilities within the shape is maximized, see Figure 4.10 (e). Finally we rendered the pixel probability with its neighbors, and propagated the contour to a finer location based on the probabilities. We set $\tau = 0$ as the threshold in step 6 of Figure 4.4 since we obtained the probability map by setting the background probability to 0 in the previous steps. If the probability on the contour is larger than the threshold, the contour expands outwards; if the probability is less than the threshold, the contour shrinks inwards. The resulting object is therefore shown in Figure 4.10 (f).

**Computational complexity**

The dominant elements approach has a lower computational cost for the involved searches since neither iterative location search nor complex object modeling is required. To briefly estimate the searching complexity, we denote by $n_f$ the number of pixels in a frame, and $n_o$ the number of pixels in an object template. In the dominant element approach, if we use $n_e$ elements to represent the object as described earlier in this chapter, and adopt the triangle function for (4.3), then the searching complexity will be $O(n_en_f)$. This means we need to compare all the pixel positions in the current frame with each element in the object template. Furthermore, if the constrained similarity function (4.5) is used
and $n_x$ pixels satisfy this condition, then we only need to compare $n_x$ pixels with each element. Hence the computational complexity reduces to $O(n_o n_x)$ with $n_x$$<<$$n_f$. In a typical template matching, the pixels of the object template have to be each compared with that of the corresponding candidate region at all positions in the current frame, such as in (2.4). The incurred comparison complexity is thus $O(n_o n_f)$. Obviously the dominant element approach requires much less number of comparisons than the normal template match since $n_e$$<<$$n_o$ and $n_x$$<<$$n_f$. The same approach can also be applied to the kernel estimator approach in chapter 2. Suppose there are $n_k$ kernel probability elements derived from the object of $n_o$ pixels in the second approach there. If we search through the whole frame, then $n_k$ kernel probability elements will have to be compared with the corresponding block of kernel elements at all positions in the current frame. Hence the computational complexity will be $O(n_k n_f)$. Since $n_k$ is always much less than the number of the object pixels $n_o$, much less number of comparisons is required in constrast to the typical template match. If one conducts a block matching of $n_k$ pixels and only $n_s$ searching steps are involved, then the computational complexity is simply $O(n_k n_s)$. This complexity is on a par with $O(n_o n_x)$ derived for the dominant element approach when $n_k$ $\approx$ $n_o$ and $n_s$ $\approx$ $n_x$. However the computational cost in obtaining the kernel probability elements in (2.7) is higher than in obtaining the dominant elements in (4.1). We finally note that this analysis did not have to take into account the object shape refinement since its effect would be the same across all the considered methods.
4.5. Summary

In this chapter we presented an object model of dynamic dominant feature elements for an efficient tracking. The color intensity bands from which the feature elements are derived are determined by their collective power to distinguish themselves from their local background. The probability rendering of pixels on the current frame can well accommodate the elements model to extract the similar object pixels. The noises are greatly reduced by a threshold or constrained similarity function. This method can also deal with partial occlusion, object internal change and illumination change due to the probability framework. The shape deformation is resolved by its prior shape and the main features captured for the element updates for the subsequent frames. The significance of the approach lies in its lower computational cost by the direct pixels matching and its adaptability to the moving background, although the object extracted may not accurately represent the real object shape in the extreme case such as when there are similar color distributions in both the object and the background.
5.

CAMOUFLAGED OBJECT

5.1. Introduction

Tracking an object through a video sequence is to locate the position of an object of interest as well as to estimate its overall occupation accurately and efficiently in the subsequent frames. Depending on the purposes of applications, the requirements for the exact description of the object appearance at any particular time may vary as long as the object’s position is obtained. Basically object tracking is classified into three categories: tracking the significant feature points of an object, tracking the boundaries of an object, and tracking the region an object occupies. Most research has been focused on the ideal situations where the object has smooth and consistent color, regular texture, or sharp differences of an object attribute compared to its background. Some algorithms developed
under such circumstances have subsequently achieved quite reasonable performances. However there are many fewer results or even attempts at the tracking problems when the object to be tracked does not differentiate itself well from the background. One of the major purposes of this chapter is to investigate the tracking problems under such challenging environment. More precisely, we will here concentrate on tracking camouflaged wild animals which has application potentials in monitoring, analyzing and protecting endangered species or the ecology as whole. We develop new methods such as weighted region consolidation for extracting an object hidden in apparently chaotic and camouflaged scenes and contour voting for trusted object boundary.

Object tracking has received active research for many years, albeit largely in welcoming environments where tracked objects distinguish themselves well in shape or color. Template matching (Jain et al 1996; Jolly et al 1996) has been widely studied and is straightforward on tracking. The underlying assumption there is that the object appearance in the frame sequence remains the same. If the illumination condition or object orientation changes, such approaches could very easily fail. An active contour model typically attempts to extract the boundaries of an object, and then tracks these boundaries through the image sequence by incorporating available information such as significant edges and curve smoothness. Such a model is based on the observation that the object boundaries should minimize a properly chosen energy functional. For tracking objects in a cluster background, the maximum a posteriori framework was proposed (Tao et al 2002) using the EM algorithm. Isard and Blake (1998) proposed the
condensation algorithm that propagates the density by maximizing the posterior probability in the cluster background. In our previous chapters, we have developed methods that focus on modeling the significant distinctive features of the object appearance. Stable appearance, similar colors and regular textures, discriminative density in local environment and spatial information incorporation have been well modeled there via a non-parametric estimation. We note that if the distinctive features are not so significant in some video scenarios, the model of such features could be less successful in helping to capture the object of interest. We are hence motivated here to investigate how to track camouflaged objects.

We recall that the active contour model (Bertalmio et al 2000) is actually also based on the assumption that the object to be tracked will exhibit strong intensity boundaries from the background. This is because it relies on the edge indicators that usually resort to the intensity gradient. Whenever the object boundaries become undecipherable or ambiguous, the tracking will be difficult to proceed. As the object color is similar to its background, the color-based approaches also have difficulties to accurately locate the object. To overcome this critical problem, we propose a method to track an object with relatively weaker boundaries due to the background noises caused by such as animal camouflage. We design and conduct proper motion detection and image noise reduction before embarking on the contour approach to track our objects. The work done here has been published in (Huang and Jiang 2005).

This chapter will be organized as follows. In section 2 we describe the
challenges this chapter aims at in the form of a concrete image sequence and develop the method of weighted region consolidation and contour evolution. In section 3, an alternative contour extraction is proposed to improve the accuracy of the contour based on a confidence voting mechanism. Section 4 contains the implementation and discussion.

5.2. Tracking camouflaged object

5.2.1 The case of challenges

While our main purpose is to investigate tracking camouflaged objects we will illustrate the difficulties or the challenges of such a generic problem by featuring a typical application scenario. For this purpose, we will initially experiment on an image sequence in which a meerkat runs from the front to the rear on a ground of similar color; see some frames in Figure 5.1. In fact part of the video clip also contains a partial occlusion of one meerkat behind a rock of similar color with another meerkat running out of the same rock towards the rear. We note that the meerkat has a strikingly similar color with its nearby background. In fact the hue is very close in most area. Other than the similar color and the random texture-like background, there is also certain sharp contrast at the far end background. This could easily result in unsteady background and large errors through the image sequence due to such as small camera shaking even though we may claim that the camera is held statically. We stress that the object of our interest in this case is non-smooth and non-rigid and, moreover, its color changes slightly during the movement because of the illumination changes. The background is also
somewhat sensitive to such changes.

The noises from the background are large compared to the object when we difference two frames. In some cases, the noises are even larger than the object itself. The noises make it difficult to segment the motion. As the result of large error in the background extracted from the sequence frames and the similar color between the object and the background, the object is difficult to effectively extract from the background. The object displays relatively weak boundaries in comparison with the background. All these characteristics pose challenges on the motion detection and on the object extraction. They also mean that the traditional methods such as block match method, background subtraction, or contour method will be largely ineffective.

![Video sequence](image)

**Figure 5.1** Video sequence

Our main strategy is thus to overcome the hurdles of these unwelcoming characteristics by reducing the background noises, identifying and using unique feature, and developing algorithms sustainable under such harsh conditions. We
will design the solution framework to be sufficiently flexible so as to allow additional control and adjustment to adapt to the variation of the problem environment and to best improve the tracking performance.

Tracking camouflaged objects is a complex problem by its own nature, and the process will thus be composed of a number of components and steps. Based on the characteristics of the image sequence of a moving camouflaged object, we break down the object tracking problem into several major steps. We first undertake to reduce the image noises to a lower level. Then we identify the motion area and refine or consolidate such an area. Global edge detection will then be utilized to support the object identification in fine-tuning the true object boundaries. We also propose to incorporate other factors such as object shapes into the energy functional. This is in principle a contour approach pivoted by strategies and methods specifically designed for the circumstances.

5.2.2 Background synchronization

For simplicity we categorize all the non-uniformity of the background shape and texture as noises in the frame sequence. Such noises can be caused by unsteady cameras let alone large camera movement. They are also sensitive to illumination variations. This turns out to be a major contribution of the inaccuracy in the background extraction, motion segmentation and object identification. The smooth low-pass filter and position adjustment are thus designed to help reduce the noise level from the background.

The smooth filter is aimed at reducing sharp contrast part of the background so
as to reduce the noises in the frame difference. The filter is constructed as a weighted averaging filter similar to (Gonzalez and Woods 2002).

The position adjustment attempts to better synchronize the current frame with the previous frame so that the inter-frame difference data is produced with less noise in the background. The process requires pixel interpolation in the current frame. By assuming the picture displacement caused by an unsteady camera is small, we calculate the actual pixel displacement \((u, v)\) with \(-1 \leq u, v \leq 1\) through minimizing the mean squares error between adjusted current (interpolated) frame and the previous frame via (2.10). In other words, we shift the current frame in the horizontal and vertical directions by \(u\) and \(v\) pixels respectively so that the displaced image better matches the previous frame:

\[
(u, v) = \arg \min \sum_{(x,y) \in \Omega} [I_p(x, y) - I_c(x + u, y + v)]^2,
\]

where \(I_c(x,y)\) and \(I_p(x,y)\) are the pixel intensities for the current and the previous frames respectively, and \(I_c^{\alpha}(x,y) = I_c(x+u, y+v)\) is given by

\[
I_c(x + u, y + v) = \left|vu\right|I_c(x + sgn(u), y + sgn(v)) + \left|v\right|(1 - \left|u\right|)I_c(x, y + sgn(v)) + \left|u\right|(1 - \left|v\right|)I_c(x + sgn(u), y) + (1 - \left|v\right|)(1 - \left|u\right|)I_c(x, y)
\]

We note that the above formula is derived from the interpolation of the four neighboring pixels surrounding the interpolated point. After calculating the optimal adjustment parameters, we can then generate the inter-frame difference data for the later steps. For a pixel displacement of more than a pixel, a similar
method can be applied to derive the proper \( I_c(x, y) \).

5.2.3 Weighted region consolidation

5.2.3.1 Motion region extraction

After the background synchronization, the object motion would be the active component among the frames. The object motion can be estimated from the interframe difference data

\[
D(x, y) = |I_c(x, y) - I_p(x, y)|,
\]

and the difference \( D(x,y) \) should reflect the change of the scenes. Due to the strikingly similar colors between the object and background, the detectable signal on the movement remains very weak. On the other hand, the object area overlapped during the motion is often visible because of the inconsistent object color changes during the object movement. Based on this observation, we propose to enhance the motion pixels with high probability on spatial and intensity density.

For this purpose, a filter that measures the spatial and intensity difference is applied on the interframe difference data to select the potential motion pixels, and it reads

\[
D^N(x, y) = \frac{\alpha}{n} \sum_{i=1}^{n} D(x_i, y_i),
\]

where \( n \) is the number of pixels associated with the current pixel within a given neighborhood, and \( \alpha \) is an adjustor. Only pixels with \( D^N(x,y) > \tau_m \), a motion pixel threshold, are regarded as potential motion pixels. A weighting mask based on
pixel distances is then used to render these potential motion pixels via

\[ D^M(x, y) = \frac{1}{n} \sum_{(i, j) \in \Omega^c} w(i, j) D^N(x + i, y + j), \] (5.3)

where \(||(i, j)||\) is the distance from the pixel at \((x+i, y+j)\) to the current pixel at \((x,y)\), \(\beta\) is an adjusting constant, \(\Omega^c\) is consolidation window, a neighborhood centered at pixel \((x,y)\), and \(n=|\Omega^c|\) is the cardinality of \(\Omega^c\). The new image \(D^M(x,y)\) then enhances the motion of the object by enhancing the weak motion pixels due to the similar color of object and background with the help of the surrounding motion pixels and the reduction on the sparse noise from background.

### 5.2.3.2 Threshold

The threshold \(\tau_m\) mentioned in the above determines the potential motion pixels. It can be selected on the basis of the distribution of quasi-mean over inter-frame difference data. For presentational simplicity and clarity we assume, without loss of generality, that the rough size of the object is known and the object movement is not too fast. By estimating the proper proportion of pixels that potentially represent the motion area, the threshold can be readily determined and can result in the segmentation of the motion.

### 5.2.3.3 Consolidation window size

Choosing a proper consolidation window size for weighted region consolidation (WRC) to check the spatial and intensity difference is critical to estimating the motion area. If an object has the same color in the overlapped part
from two frames, the motion information is weak there. If too small the size of window could result in splitting the motion area. If the size of window is unnecessarily large, non-motion areas could be generated. Consequently, the size of the consolidation window is determined by the object’s characteristic size and color distribution. In general, the larger the object size and the areas of consistent color in the object, the larger the consolidation window.

5.2.3.4 Iterative processing

After applying the mask on the inter-frame difference data, a rough motion area is obtained. Next we apply the region consolidation process via (5.3) and (5.2) iteratively to enhance the motion region. A new threshold is determined to obtain potential motion pixels based on the distribution. The consolidation window size for (5.3) and the masking window size for (5.2) may be kept the same or adjusted. The iteration will be terminated by an accuracy threshold $\tau_a$. Threshold $\tau_a$ is the minimum number of motion pixels to be created during the current region consolidation iteration. In other words, the iteration terminates when the consolidated region becomes reasonably stabilized.

If the final outcome from the iteration of the WRC contains several regions, they may be either object parts caused by a small consolidation window or a fast movement without overlap or background regions created by large background noises. When we compare them with those from previous frames, the background regions can be identified if it has the similar position and exhibits drastic change in the size of the area due to the different threshold chosen for the intensity difference in the WRC process. In other words, the motion can still be identified
by referring to the previous motion.

5.2.4 Refine object region

The motion region derived from the previous section in fact includes the object positions at the two successive frames and is thus larger than the area of the object to be detected in the current frame. By making use of the largely similar region consolidation method, we in this section describe how to further refine the region to reflect the object location in the current single frame.

We first mask the inter-frame difference data \( D(x, y) = I(x, y) - I_p(x, y) \) by motion region \( \Omega \) obtained in the previous section, and denote the result by \( D_\Omega(x, y) \). Next we select to keep the positive values of \( D_\Omega(x, y), D_\Omega^+(x, y) \), which basically contains the object information in the current frame. Then \( D_\Omega^+ \) is multiplied by a factor \( \eta \) to strengthen the pixel values before being added back to the original motion representation \( D_\Omega(x, y) \). This then leads to the area data \( D_\Omega'(x, y) \) enhanced for the object in the current frame

\[
D_\Omega'(x, y) = \eta D_\Omega^+(x, y) + D_\Omega(x, y). \tag{5.4}
\]

Finally we can apply the iterative method of region consolidation again over the \( D_\Omega'(x, y) \), and derive the object location in the current frame. The new threshold \( \tau'_m \) is thus selected on the basis of the distribution of the quasi-mean over \( D_\Omega'(x,y) \). We note that the background at this stage has already been excluded to speed up the process, and this makes it possible to enhance the object in the current frame without the degeneration or interference by the background.
noises.

If we, for clarity, convert $D'_\Omega(x, y)$ into a binary image, then the boundary of the current object can be obtained by first eroding $D'_\Omega(x, y)$ by a 3x3 structuring element, and then calculating the difference between $D'_\Omega(x, y)$ and its erosion (Gonzalez and Woods 2002). As the object region is obtained by the rendering, it is a bit larger than the exact object boundary. Yet it maximally reduces the background effect, as showed in the edge map in Figure 5.7. This boundary can be subsequently treated as the initial contour for the next contour propagation to further refine the object shape.

5.2.5 Shape Integration

As the background is cluttered and the colors of the object and the background are very similar, the edges obtained by such as Canny detector is cluttered or lost after thresholding, as shown in Figure 5.7 (a). Also the object is not distinguishable from the background since their colors are not much different. It is difficult to capture the object accurately by relying on only one piece of insufficient information such as object color or edges. We need to make use of all useful available information. Except for the motion information which may be estimated as above, and the edge information from the standard methods, the object’s shape information in the previous frame should not be ignored as it complements the contour refinement. A prior knowledge of the object shape could be defined and assumed to be known. Based on the assumption of shape similarity of consecutive frames, the shape may be used on the current frame to
synchronize with the object contour from the motion segmentation. We propose to use significant feature points to match the shape to the corresponding locations by referring to the edge map along with potential adjustment. Though the object is a non-rigid object, its shape information still helps refine the object from its immediate subsequent frame. The weight factor contributing to the extraction depends on the deformability of object.

5.2.6 Contour evolution

The proposed iterative region consolidation will, in general, result in a reasonable contour for the tracked object although the borders may be somewhat slack. This will not at all affect the fact that the object of interest is being properly tracked by a sufficiently conclusive contour because the deviation of the slack edges from the true object boundary is of a small scale. In other words, the object tracking can be conducted by, and largely based on, the algorithm of WRC and its variants. If, however, a further refinement on the contour is desired then the contour evolution through the use of an energy objective function can be performed to improve the accuracy of the object boundary. In this regard, the object’s contour will start from the initial contour obtained from motion segmentation or its expanding contour and then evolve towards a better boundary by minimizing an energy function which integrates with the object edges and shape information. For this purpose, we will make use of an edge indicator and will take into consideration the consistency of the neighborhood pixels both on and off the edges. For the portions of edges that may be apparently absent from the edge map, its derivation will depend on the smoothness and the shape
information. Hence our energy function for the contour evolution is proposed to be

\[
E(c) = \int_{0}^{L} [\alpha g(\nabla I(p)) + \beta \vec{g}(\nabla \bar{I}(p)) + \gamma \nabla s^e(p) \psi(p) + \delta \vec{c}(p)] dp,
\]

where \(g(\cdot)\) and \(\vec{g}(\cdot)\) are edge indicators, which have higher values on the edge and lower values in a smooth area, \(\bar{I}\) is any kind of mean image from the original image \(I\), \(\nabla s\) is the distance from the current pixel to the nearest edge in the edge map or to the shape established in the previous frame depending on a measure \(\psi\). The last term is a smoothness attribute of the contour. The coupling factors \(\alpha, \beta, \gamma\) and \(\delta\) can be carefully selected to place different significance on the specific energy terms. The accuracy of contour, for instance, may subsequently increase or decrease over the change of parameter \(\gamma\), depending on how weak the boundary is.

The approach is focused on pieces of contour. The nearest edge on the edge map for current pixels is decided by detecting the nearest group of pixels with large values, ideally they should be close and somewhat in parallel to the existing initial contour. The algorithm will minimize energy for the edge pieces that are close to the initial contour and the standard deviation of the distances is small. We can then propagate the contour by linearly transforming towards the matched edge pieces.

To summarize all the undertakings in this section, the main procedure involved in the tracking is shown in the following Figure 5.2.
The object boundary can be refined by the evolution of the contour through minimizing an energy function incorporating motion detection, edge and shape information in the above for the purpose of extracting a more accurate contour from a collection of insufficient information piece coming from such as a camouflaged object. However this means the balancing coefficients in the energy terms have to be carefully selected. Alternatively, the contour evolution can be achieved through a different approach, the confidence voting based on the trusted information available. The contour will be evaluated with the combination of the confidence assigned to it by the algorithm that generated it. In the following, we will describe in detail a procedure with a broader view to it becoming a general framework for contour evaluation rather than being limited to the consideration of camouflaged objects.

Figure 5.2 Contour extraction

5.3. Contour voting

The object boundary can be refined by the evolution of the contour through minimizing an energy function incorporating motion detection, edge and shape information in the above for the purpose of extracting a more accurate contour from a collection of insufficient information piece coming from such as a camouflaged object. However this means the balancing coefficients in the energy terms have to be carefully selected. Alternatively, the contour evolution can be achieved through a different approach, the confidence voting based on the trusted information available. The contour will be evaluated with the combination of the confidence assigned to it by the algorithm that generated it. In the following, we will describe in detail a procedure with a broader view to it becoming a general framework for contour evaluation rather than being limited to the consideration of camouflaged objects.
5.3.1 Voting confidence

A rough contour may be satisfactory for tracking an object, especially when the needs well fit with the used algorithm, while in other cases, a more accurate contour for the object may be required for better understanding the object behavior. Also, the object may need to be verified since the performance of most methods is subject to the images’ environmental conditions. Though some approaches take certain properties such as geometric information or image features into account in their choice of the energy functions, it is still largely the result coming from a single particular approach rather than from other independent verifications as well. We thus propose to evaluate and synthesize the contours, or even partial contour segments, generated by different approaches and to estimate a more accurate contour by measuring the closeness of the different contour segments and the corresponding segment reliability determined by how much the local image area of the segment fulfils the algorithm requirements.

The voting scheme aims to improve the accuracy of the object boundaries. We note that how to assign confidence to sections of boundaries is typically linked to the process of deriving them in the first place.

We know there are a few ways to obtain object contours in literature. The active contour model is, for instance, a well known approach that usually takes the intensity changes as the main criteria together with some other properties. Therefore this contour approach may be inaccurate in non-strong boundaries. Also it can’t handle the image shade problem and may be interrupted by background noises. Moreover, smoothness is often imposed on the contour while
it propagates so the contour obtained may not accurately reflect the object boundaries. The edge map on the other end usually provides overwhelming edge information including object contour, internal edges as well as background edges. In (Nguyen et al 2002), the authors there made use of predicted motion from the block match to remove background edges by projecting the previous frame onto the current frame. Earlier deformable template approaches have tried to evaluate the shape change of the template with the successive frames. However, these approaches can not measure the extent the obtained contour reflects the real object by itself. We thus propose a voting scheme by assigning confidence, a value between 0 and 1, to the contour segments to reflect how much we trust the contours obtained. The final step is to make use of the best estimation among contour segments obtained from different sources and approximate the contour as accurately as possible.

Suppose we have \( n \) contours or even partial contours, \( \{\Gamma_i\}_{1 \leq i \leq n} \), derived from a few different approaches, and each \( \Gamma_i \) contains \( m_i \) segments with the corresponding confidence \( \rho_i^{(k)} \) for \( k=1, \ldots, m_i \). The synthesis of these contours will be actually performed pair-wise on, say \( \Gamma_i \) and \( \Gamma_j \). If a segment in \( \Gamma_i \) overlaps partially over another segment in \( \Gamma_j \), then the synthesis will be considered for this pair of segments and will start from the pixels in the overlapped portion. If no segment in \( \Gamma_i \) has intersection with \( \Gamma_j \), then we search for its closest segment in \( \Gamma_j \). If we assume that such closest distance is between a point \( v_i \) in the segment in \( \Gamma_i \) and a point \( v_j \) in the segment in \( \Gamma_j \), then the synthesis point will locate between \( v_i \) and \( v_j \), i.e. \( d_i = \lambda d \) and \( d_j = (1- \lambda)d \) with \( 0 \leq \lambda \leq 1 \), \( d \) is the distance between \( v_i \) and
\( v_j \), and \( d_i \) is the distance from \( v_i \) to the synthesis point. We start from this point and proceed to draw the synthesized contour pixels, as shown in Figure 5.3, on the basis of individual segment confidence, the total confidence and confidence differences. If we model the confidence via a triangle function, the \( \lambda \) and the new confidence \( \rho \) of voted or synthesized point will then be given by

\[
\lambda = \frac{\alpha (1 - \max(\rho_i, \rho_j))}{1 - \min(\rho_i, \rho_j) + \alpha (1 - \max(\rho_i, \rho_j))},
\]

\[
\rho = \max(\rho_i, \rho_j) + \lambda (1 - \max(\rho_i, \rho_j)) / 2,
\]

where \( 0 \leq \alpha \leq 1 \) controls how much the synthesized point should be inclined towards the segment of higher confidence although \( \alpha = 1 \) implies no bias. The synthesized point locates at \( \lambda d \) to the point with higher confidence. More precisely, we first set two confidence thresholds \( \mathcal{L}_h \), over which the contour segment would be completely trusted, and \( \mathcal{L}_l \), under which the segment would be considered useless. Then we will make use of (5.5) to determine the synthesis point unless either the difference of two confidences is larger than a threshold \( \mathcal{L}_d \) or the confidence of both segments is smaller than \( \mathcal{L}_l \). For the first case the pixel with higher confidence is trusted as a true contour point. For the second case, if both confidences are lower than \( \mathcal{L}_l \) and the sum of the two confidences is also low, e.g. lower than a threshold \( \mathcal{L}_e \), then this point is considered uncertain and should be interpolated by its neighbours. For instance, one can follow the previous known synthesized contour direction, or evaluate again with an additional method subject to possible adjustment later on.
As $v_i$ moves along $\Gamma_i$, the paired $v_j$ will be chosen so that it is the closest point in $\Gamma_j$ to $v_i$. We therefore repeat the above procedure and derive the next synthesized point. If the spatial distance between current synthesized pixel and its predecessor is larger than two pixels, which may happen when a new confidence segment is involved from one of contours, we can interpolate or fill up the discontinued segments in a later step.

The search window for closest and significant pixels in $\Gamma_j$ will affect which pixel to be chosen for the voting synthesis. If the window is too large, the pixel found may be misleading. However too small a search window could exclude some potentially good candidates for the voting scheme. Empirical results show that a suitable sized search window would just cover the biggest distance of two contour segments.

The thresholds $\mathcal{L}_h$ and $\mathcal{L}_l$ are set according to the reliability of the algorithm for the contour. For example, if we believe the contour from an edge map is reliable due to the image nature the value $\mathcal{L}_h$ can then be set so that the contour will include a reasonable number of edges from the edge map. The adjusting
factor $\alpha$ can also be chosen to incline more towards the reliable contour segment.

Hence the refinement of an object contour is achieved through the voting process in terms of the confidence of different contours. For example, if we have a predicted motion contour we may verify and obtain a better contour with an edge map by voting. If a contour from an active contour model includes a shade area, it is possible to pull the contour towards the object by the edge contour which is supposed to be obtained with predicted object shape. The accuracy of the synthesized contour largely relies on the available contours of proper confidence. In the following subsection, we describe the mechanism of confidence assignment for the typical contours or contour segments.

5.3.2 Confidence assignment

Many algorithms are developed on specific scenarios with limitation assumptions. The confidence assigned to an object contour can be based on the characteristics of the algorithm and the extent the limitation assumptions are satisfied. The confidence assignment may be carried out in the process of obtaining the contour by examining the satisfaction of the algorithmic assumptions or by using an additional method suitably tailored for the process. We thus developed, in particular, the confidence assignment mechanism for contours obtained through motion detection, active contour from the snake model, and edges from edge maps. The confidence will range from 0 to 1, and a higher value means higher reliability.
5.3.2.1 Motion and object detection

We illustrate the confidence assignment for the contour obtained from motion information using the proposed method in this chapter. When objects move in a scene in a static background or an unstable background caused by an unsteady camera, illumination change, or sampling error, the motion detection can be achieved via inter-frame difference data. The successive frames are synchronized by adjusting the current frame to the previous frame subject to an affine transformation. Though we may still obtain a noisy background we can obtain a contour $\Gamma'_d$ by applying the weighted region consolidation method as described in the last section on the inter-frame difference data. We group the potential motion pixels by their spatial and intensity difference, $D(x, y)$, and rebuild these pixel values with its neighbours within certain distance via (5.3). Only pixels with $D^M(x, y) > T_m$, a motion pixel threshold, are regarded as potential motion pixels. After an iterative process for the smooth motion, the object region is then obtained. The motion contour $\Gamma'_d$ is obtained at the end after eroding on the refined region.

If we want to refine a contour further, we can reuse the inter-frame difference data, along with enhanced current object information, to identify in fact the current object from motion detection image obtained from above by the same method. The weighted region consolidation method employs a chosen threshold which affects the accuracy of the motion region or object region, and an unsuitable threshold or size of mask may result in too large or too small a region. Thus the confidence assigned to the contour pixels can be based on the
characteristics of the local probabilities which are obtained by the weighted region consolidation. For instance, if we calculate the sum of local probabilities for the contour pixels, i.e. \( p' = \frac{1}{n} \sum p(x, y) \), within a local window containing \( n \) pixels centered at \((x, y)\), we can allocate high confidence to contour pixels with high probability \( p' \). The value of confidence can adopt the probability \( p' \) directly or rescaled accordingly.

### 5.3.2.2 Active contour

With an active contour model, where a contour is controlled by an energy function such as in (Paragios and Deriche 2000) using intensity change and smooth property, we can obtain a contour \( \Gamma_a \). Basically, if the object is in a smooth background, therefore less interrupted by the background, we could assign a relatively high confidence to the contour \( \Gamma_a \). We divide the contour into segments based on the similarity of the intensity changes on both sides along the contour; each is assigned with the corresponding confidence from 0 to 1 according to their strength of intensity changes, as shown below in (5.6)

\[
\Gamma_a = \bigcup \ell_k, \quad \ell_k = \{(x_j, y_j)\},
\]

\[
|\nabla I(x_j, y_j)| - \mu_{\ell_k}(|\nabla I|) < \tau_k,
\]

\[
\rho^{(k)}_a = \frac{\alpha \mu_{\ell_k}(|\nabla I|)}{\sum_{d=1}^{m} \mu_{\ell_d}(|\nabla I|)},
\]

where \( \ell_k \) is the \( k \)-th segment of contour containing “connected” \((x_j, y_j)\), \( I(x, y) \) is the image intensity, \( \mu_{\ell} (\cdot) \) denotes the average intensity change on the two sides of the segment \( \ell \), \( \tau_k \) is a threshold that determines how similar the intensity change of pixels in a segment are, \( \rho^{(k)}_a \) is the confidence of the \( k \)-th segment, and \( \alpha \) is used to adjust the importance of this contour to those from the other sources.
5.3.2.3 Edge-based contour

With an edge detection function such as Canny detector, an edge map can be generated. The edge map is further refined by suppressing the weaker and dispersed edges. Then we need to identify candidate edges for the contour. This can be achieved with the help of projected object shape from the previous frame.

We match the significant (strong and continuing) edge segments from an edge map to the shape by maximizing the similarity of the edges. Then the overlapping significant edges are assigned with higher confidence. Along the shape, we search the significant edges near the shape and assign relatively high confidences accordingly, such as a value derived from the edge. We also assign confidence to less significant edges nearby with a lower value, which may be potential edges for the contour in some special cases. The larger the distance is to the shape, the smaller the confidence is to be assigned. For instance, the confidence can be calculated simply by $\rho = \alpha z / d$, where $z$ is the edge strength, $d$ is the distance to the shape, and $\alpha$ again is used to adjust the importance to the other contours from other sources. Thus we obtain a candidate edge contour for the object, denoted as $\Gamma_c$. For the projected shape itself, the confidence is the same along the shape unless there is a prior knowledge to help identify the difference, such as if the partial contour is known to be highly possible to undergo deformation, in which one can set a lower confidence.

5.3.3 Voting evaluation

The contours obtained by the above methods now have an extra property -
confidence, which indicates the reliability of each contour segment. We evaluate each segment from different sources to obtain an optimally synthesized contour segments using the method described in section 5.3.1. Yet the segments from different sources are not exactly peer to peer correspondence and the confidence may vary significantly when it is close to its neighbour segment. As a result, the synthesized segment may not be smoothly connected. The final step is thus to smooth the disconnected segments.

For each new contour segment, there is a new confidence calculated from its original segments. We start from a segment with high confidence, and then extend it towards the lower confidence segment nearby if pertinent. One simple way is to determine a connection point in the lower confidence segment so that the connection will be smooth, add new pixels by a transform such as linear transform between the higher confidence pixel and the connection point, until the pixels are connected from the two segments. To summarize, the main steps to synthesize $n$ contours are:

i) Assign confidence $\rho$ to contour segments in the process of obtaining contours $\Gamma_1 \ldots \Gamma_n$ based on certain criteria, as illustrated above for motion estimation, active contour model or edge maps.

ii) For a chosen pair $\Gamma_i$ and $\Gamma_j$ and for a given pixel along $\Gamma_i$, find the corresponding closest pixel in $\Gamma_j$ within a search window, if no such pixels are available, then go to the next pixel in $\Gamma_i$.

iii) Obtain the synthesized pixel location and new confidence by voting via the
confidence of the chosen pair of pixels from $I_i$ and $I_j$. Then move to the next pixel in $I_i$ and repeat step ii.

iv) After all pixels in $I_i$ have been evaluated, a synthesized contour $\Gamma_{syn}$ is obtained. Interpolate or smooth the disconnected segments with the priority given to contour segments of high confidence.

v) Choose next contour $\Gamma_{j+1}$ if available, repeat the process from step ii to iv for $\Gamma_{syn}$ and $\Gamma_{j+1}$, a new synthesized contour is obtained. Then repeat this process until no further contours are left to be synthesized.

5.4. Implementation and Demonstration

5.4.1 Camouflaged object

We implemented the proposed weighted region consolidation method with real video as shown in Figure 5.1, the size of frames is 343x276 pixels. The filter size for background is 3x3 pixels. We choose for each inter-frame difference data a 5x5 pixel neighborhood for (5.2) to determine if the pixel needs to be consolidated with (5.3), and then choose a 7x7 pixel consolidation window to enhance the pixel via (5.3) with $\beta=1$. In the subsequent iterations via (5.3) we choose a larger window size of 9x9 pixels since more information may be desired in the later iterations. For the use of (5.2) we took $\alpha=25/9$, bearing in mind that $n=25$ there. The accuracy threshold $T_a$ for the termination of the iteration discussed in section 5.2.3.4 is taken to be 20, which means the process would stop if the difference of the results from two successive iterations is no more than
20 pixels. The $T_m$ in section 5.2.3.2 was chosen based on the inter-frame difference data. For instance, the data may be grouped into ten bins and the $T_m$ takes the value so that the corresponding difference of its successive bins is larger enough. In other words, it is the value that can lead to roughly the number of pixels in the estimated area. In this implementation, the contour refinement via $\nabla$ is based on an edge map such as those shown in Figure 5.7. Figure 5.4 shows the results from 1st to 4th iteration via the process of WRC. The background noises are reduced and the motion region is also refined iteratively.

![Figure 5.4 Motion region segmentation](image)

In Figure 5.5, the potential motion pixels are illustrated in (a) via (5.2), and for clarity the remote background noise is not shown here. The enhanced object pixels within the motion region are illustrated in (b) via (5.4). Therefore the object pixels are enhanced from the motion region. For an intuitive comparison, we projected the contours obtained from different steps in the original frame as shown in Figure 5.6, where (a) shows the extracted motion region in the previous frame and (b) shows the motion region in the current frame. The motion region
covers both the object shape at different time stamps. After the object region was refined, a more accurate contour was then obtained. Figure 5.6 (c) shows the object region in the previous frame and Figure 5.6 (d) shows the object region in the current frame. We can see that (d) has the best contour fitting there.

![Figure 5.5](image)

**Figure 5.5** (a) Interframe difference data. (b) Enhanced current object.

![Figure 5.6](image)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

**Figure 5.6** (a) Motion region in previous frame. (b) Motion region in current frame. (c) Object region in previous frame. (d) Object region in current frame.

For further improvement, the previous object shape was incorporated in the contour evolution. Figure 5.7 shows the object location in the edge map. Figure 5.8 shows the final contour that is evolved from the energy function leading to the more accurate boundary for the current object.
The sequential tracking process is finally demonstrated in Figure 5.9. From left to right, top to bottom, the outlined objects are the tracking results from the frame 2nd, 3rd, 4th, 5th, 6th, 18th, 23rd, 35th, 131st respectively. They give us a global view. The smaller the object, the less accurate the contour is because the mask and consolidation window size have not been adjusted.

The choice $T_m$ could significantly impact on the resulting region and the convergence speed of the iteration process. If $T_m$ is small, the derived region may be larger and therefore with more background noises included. If $T_m$ is large, then the derived region may be too small to completely cover the real motion region, or even result in some holes within the region. A larger consolidation window size avoids missing the potential pixels but could also lead to some other
unwanted pixels. The size of the window thus determines the number of pixels that contribute to the new rendered pixel values.

![Figure 5.9](image)

**Figure 5.9** Object captured from 2nd, 3rd, 4th, 5th, 6th, 18th, 23rd, 35th, 131st frame respectively.

The tracking efficiency can be adjusted via the relevant thresholds and the size of mask or windows. In our experiments most iterative processes were completed in only very few iterations. Also once the area was roughly identified the iteration process can be more locally performed.

### 5.4.2 Contour evaluation

For clarity we used another sample video to illustrate the voting process. The problem for a real video sequence is complicated as the contours may be fragmented and multiple candidate edges may be available nearby in the same frame. Figure 5.10 illustrates how a contour from the motion detection approach
was both verified and refined via an edge map. Here the motion contour was assigned with an equal confidence of 0.5. In the edge map the edges were assigned with different confidence. For example, after the noise removal via threshold in Figure 5.10 (c), confidence of 0.9 was assigned to significant edges and 0.7 was assigned to the dispersed edges (less than 6 pixels connected). The search window for the closest pixels in the edge map is 22 pixels wide, and the closest pixels found in edge map are shown in Figure 5.10 (d). After the voting and synthesizing, the new contour pixels were obtained and shown in Figure 5.10 (e), depicted in dark dotted lines, and the smoothed new contour is shown in Figure 5.10 (f). It shows the contour is closer to the edges. The position of the new contour depends on the trust on the available information. If we give more confidence to the motion contour, the new contour will get closer to the motion contour.

Figure 5.11 shows the contour voting process from the motion contour, active contour and edge contour. In other words, the final contour was synthesized from these 3 individual contours. The active contour, from active contour model by intensity change and smoothness, was obtained as in Figure 5.11 (a), and the confidence was assigned to the segments according to their strength of intensity change on both sides of the contour. Figure 5.11 (b) then displays the contours from motion and active contour model on the edge map. The voting process is conducted as follows. First the new contour was voted between motion contour and the active contour, as shown in Figure 5.11 (c). Then the resulting contour votes with the edges on the edge map, as shown in Figure 5.11 (d). The resulting
Figure 5.10 (a) $\Gamma_d$ projected on the current image. (b) $\Gamma_d$ projected on an edge map. (c) $\Gamma_e$ by background noise removal from (b). (d) The closest points in edge map corresponding to $\Gamma_d$. (e) Synthesized contour voted by $\Gamma_d$ and $\Gamma_e$. (f) The smoothed resulting contour.

Figure 5.11 (a) $\Gamma_a$ projected on edge map. (b) $\Gamma_a$ and $\Gamma_a$ on edge map. (c) Synthesized contour voted by $\Gamma_a$ and $\Gamma_a$. (d) Synthesized contour voted by $\Gamma_a$, $\Gamma_a$ and $\Gamma_a$. (e) Optimal contour on current image.
synthesized contour was finally highlighted in the original image in Figure 5.11 (e). We note that the contour in Figure 5.11 (d) is similar to that in Figure 5.10 (f). They basically better reflect the shape of the object.

5.5. Summary

We proposed a weighted region consolidation method to track camouflaged objects within a noisy environment of similar colors. The motion area is extracted with the consideration of surrounding motion pixels and the object is enhanced and extracted within the motion region and with the use of the WRC method. The candidate region for the object is refined by the active contour model integrated with the shape information and the edge map. The tracking accuracy can be controlled via the thresholds made during motion pixel estimation and rendering process. The thresholds will depend on the objects, but the participation of a user in the determination of the thresholds may reduce the automation of process. In addition, a voting scheme as a general framework to verify and refine the contours is suggested for uncertain contours as in the case of camouflaged objects. It refines the contours derived from different algorithms of varying degree of accuracy and therefore can help solve the shade problem and improve the accuracy of the final object contour. In doing so it also conveniently verifies the contour in the process.
6.

WAVELET BASED IMAGE WATERMARKING

6.1. Introduction

Watermarking is to insert certain user defined information into an image or video for copyright protection and ownership authentication. The robustness is the main concern of the watermarking system, which is the ability to resist normal image processing such as geometrical operations and image compression. Invisibility is a requirement for the invisible watermarking which has a wide range of applications in which the quality of the image or video should not visibly degrade after embedding the watermark. Another important requirement is the security, an additional protection on the watermark itself. A watermark should avoid being easily exposed to the general users so as to reduce the chances of being destroyed by some of them. Based on the requirements of a watermarking system, the design of
the system usually involves several aspects, including the locations for holding the watermark, the workplace for the embedding operation, the strategy for formatting the watermark, the method for inserting the watermark, and the detection process for the watermark. Watermarking workplace can vary from the spatial domain to the frequency or spectral domain, and watermarking on the frequency or spectral domain in general provides better robustness in resisting certain geometric attacks and reducing the effect of lossy compressions.

Compared with the frequency transforms such as Fourier transform and Discrete Cosine transform, wavelet transform has shown its advantage of multi-resolution feature in the watermarking process. Most algorithms in the literature have chosen to insert the watermark in the form of a logo, pattern, or sequential bits, which can be displayed on the watermark extraction. In this chapter, however, we propose a different approach that embeds the watermark information into the decomposition process of an image where the image information is also used to fine tune the watermarking and detection process (see the related work in (Huang and Jiang 2008)). In the following we first introduce the wavelet multi-resolution for the image decomposition and the corresponding synthesis. We then develop a novel approach for watermarking based on the use of multi-class wavelet filters, and a special case of orthogonal wavelet filters for parametric watermarking is then briefly exemplified. Finally image synchronization with a part of the watermarked image is also studied for the more involved watermark detection.

6.2. Wavelet system
6.2.1 Wavelet function

Wavelets are mathematical functions that can cut up data into different frequency
components (Graps 2003). The wavelet transform not only provides the frequency
analysis, but also provides scale analysis. It allows data processing in a small scale
as well as in a larger scale. In the 1980s wavelets were introduced to digital signal
processing. A wavelet system can be generated from a single scaling function or
wavelet by scaling and translation. The base functions $\psi_{j,k}(t)$ are obtained from the
mother wavelet function $\psi(t)$ by (Burrus et al 1998)

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k), \quad j,k \in Z,$$

where $Z$ is the set of all integers, $k$ indicates time or space location, and $j$ indicates
the frequency or scale. The wavelets are hence generated by scaling the mother
function $\psi$ with $j$ and $k$. The wavelet gives a time-frequency localization of the
signal which means most of the energy of the signal can be well represented by a
few coefficients. The lower resolution coefficients can be calculated from the higher
resolution coefficients through the use of an analysis filter.

A significant aspect of the wavelet technique is its ability to provide multi-
resolution for image representation. The wavelet filter banks are capable of
decomposing an image into multi-scale so that different frequency ranges at the
different resolution levels can be obtained. The decomposition process is achieved
by the analysis filters while the reconstruction process, which converts the
frequency bands back to the original, is done by a synthesis filter. For two channel
orthogonal filter banks, the analysis filters (Dugelay and Roche 2000)
\[ c_{j-1,k} = \sum_{n \in Z} h_{n-2k} c_{j,n}, \]
\[ d_{j-1,k} = \sum_{n \in Z} g_{n-2k} c_{j,n}, \quad j, k \in Z, \]

can recursively decompose the signals, as shown in Figure 6.1.

![Signal decomposition](image)

**Figure 6.1 Signal decomposition**

Then the synthesis filter

\[ c_{j,n} = \sum_{k \in Z} h_{n-2k} c_{j-1,k} + \sum_{k \in Z} g_{n-2k} d_{j-1,k}, \]

recursively synthesizes them back to the original signal as shown in Figure 6.2.

Signal decomposition is largely a down-sampling process, and each decomposition of a selected frequency band results in two frequency components; smooth and detail components respectively. The image multi-resolution provides different spatial scales as well as frequency ranges for image analysis and operations. Moreover, the frequency bands can be reconstructed to higher resolutions by synthesis filters in the reverse order of the decomposition and all the way to the full size of the original image. The orthogonal filters and bi-orthogonal filters are both known (Burrus *et al* 1998) to provide such perfect reconstruction.
6.2.2 Wavelet filter construction

6.2.2.1 Orthogonal wavelet

The orthogonal wavelet filters can be used to decompose an image into multiple levels of bands via the analysis filters (Burrus et al 1998, Jiang and Guo 2003)

\[ c_j = \sum_{k \in Z} h_{k-2j} x_k, \]
\[ d_j = \sum_{k \in Z} (-1)^k \bar{h}_{1-k-2j} x_k, \]

and to reconstruct the bands recursively in the reverse order through the synthesis filters

\[ x_k = \sum \bar{h}_{k-2j} c_j + \sum_{j \in Z} (-1)^k \bar{h}_{1-k-2j} d_j, \]

where Z again denotes the set of all integers. For a given set of filter coefficients \( \{h, \bar{h}\} \), (6.1) uniquely characterizes the subbands of each resolution level.

If we define \( A(z) = \sum_{k \in Z} A_k z^{-k} \) with \( A_k \) being a 2x2 matrix \( A_k = \begin{bmatrix} h_{2k} & h_{2k+1} \\ h_{1-k-2k} & -h_{2k} \end{bmatrix} \),

then \( A(z) \) induces orthogonal wavelet filters if, and only if, \( A(z) \) admits the following factorization (Burrus et al 1998)
\[ A(z) = z^d R(\theta_0) R(\theta_1) \ldots R(\theta_q), \quad (6.3) \]

where \( \theta = \pm 1, q \geq 0, d \in \mathbb{Z} \text{ and } \theta_0 + \theta_1 + \ldots + \theta_q \equiv \pi/4 \text{ (mod } 2\pi) \). The orthogonal wavelet filters can therefore be generated by the \( \theta \)'s, and the \( \theta \)'s can also be utilized to process such as the carried watermark information.

### 6.2.2.2 Bi-orthogonal wavelet

For an image represented by \( \{x_k\}_{k \in \mathbb{Z}} \), suppose the \( \{h_k\} \) and \( \{\overline{h}_k\} \) satisfy the biorthogonality condition (Burrus et al 1998)

\[ \sum \overline{h}_k h_{k+2j} = \eta_{j,0} \]

(6.4)

where \( \eta_{j,0} \) is the Kronecker delta symbol, and \( j \in \mathbb{Z} \). The filter coefficients \( \{h_k\} \) and \( \{\overline{h}_k\} \) will generate two subbands \( \{c_j\} \) and \( \{d_j\} \) via (6.1), and the subbands can be synthesized by the synthesis filter via (6.2). The biorthogonal filters will become orthogonal when \( h_k \equiv \overline{h}_k \) holds for all \( k \). As more concrete examples, for any positive integer \( K \) suppose a pair of filters of linear phase has an analysis filter with length \( 2K+1 \) and a synthesis filter with length \( 2K-1 \). For \( K = 3 \), the coefficients of the corresponding 7/5-tap filters are thus determined by the wavelet biorthogonality
\[ h_0 h_i + 2 h_i h_i + 2 h_2 h_2 = 1, \]  
\[ h_0 h_2 + h_i h_1 + h_2 h_0 + h_2 h_1 = 1, \]  
\[ h_2 h_2 + h_2 h_1 = 0, \]  
\[ h_0 + 2 h_i + 2 h_2 + 2 h_3 = 2^{1/2}, \]  
\[ h_0 + 2 h_1 + 2 h_2 = 2^{1/2}. \]  

This set of equations has a few free parameters which can be used to carry a watermark or to allow other conditions to be imposed for the refinement of the filters, see (Huang and Jiang 2003; Huang and Jiang 2003)

### 6.2.3 Moments

The quality filters will result in better separation or decorrelation of the details from the smoother components of the image, leaving filtered bands more generically independent of the others and thus causing fewer reconstruction errors. This is known to be very useful in image compression. If watermarking adheres to this it will improve robustness at least to wavelet-based compression. Some of the wavelet properties, such as the consecutive vanishing moments, will result in better decorrelation for the image data. Hence, one way to obtain quality filters is to ensure a maximum number of consecutive vanishing moments.

Let \( A_k = \begin{bmatrix} h_{2k} & h_{2k+1} \\ g_{2k} & g_{2k+1} \end{bmatrix} \) and likewise for \( \bar{A}_k \), the discrete vanishing moments are defined by (Burrus et al 1998),

\[ u_r^{(0)} = \sum k^r h_k, \quad u_r^{(1)} = \sum k^r g_k, \]  
\[ \bar{u}_r^{(0)} = \sum k^r \bar{h}_k, \quad \bar{u}_r^{(1)} = \sum k^r \bar{g}_k, \]  

182
for integer \( r \geq 0 \). For the orthogonal filters, as mentioned previously, the parameters then can be chosen from \( \theta' \)'s to ensure the vanishing moments are as high an order as possible after imposing certain conditions from (6.6) readily. As an example, if we impose on the system (6.5) the vanishing moment \( \mu_0^{(1)} = 0 \),

\[
\bar{h}_0 - 2\bar{h}_1 + 2\bar{h}_2 = 0,
\]

then the solution of (6.5) and (6.7) subsequently reads

\[
\begin{align*}
\bar{h}_1 &= 2^{1/2}/4, \\
\bar{h}_2 &= -2^{1/2}h_2 / 4h_2, \\
\bar{h}_0 &= 2^{1/2} / 2 - 2\bar{h}_2, \\
h_1 &= -(2h_2^2 + 5h_2h_3 + 2h_3^2 - 2^{1/2}h_3)/(h_2 + 2h_3), \\
h_0 &= 2^{1/2} - 2(h_1 + h_2 + h_3).
\end{align*}
\]

There are two free parameters \( h_2 \) and \( h_3 \), which can be used to embed the watermark. As such, we could potentially use one free parameter to embed the watermark and use the other free parameter to carry a private key or further watermark bits. If the filter length increases, then more vanishing moments may be imposed to improve the filter quality. It is thus anticipated that further improvement can be achieved on the filters, which will in turn reflect on the overall watermarking performance of our proposed scheme.

### 6.3. Multi-class filters based watermarking

We propose a watermarking scheme based on the wavelet filters where the watermark bits are encoded in the combination of filters optimally selected for the subband decomposition process. The collection of candidate filters is established so
that all the filters within can lead to the subbands that distinguish each other to an acceptable degree. Partial coefficients, selected from the transform domain according to a user defined strategy, are extracted from one or multiple subbands and are sorted to mark the watermark existence. The watermark detection is thus to examine how well the resulting selected coefficients from the suspected image are in a sorted order as well as its monotonic increasing trend. Furthermore, watermarking performed on sub-images may offer better watermark security and provide a way to regionally watermark images or video frames. The watermarking scheme proposed here consists of wavelet filters selection, filter set construction, watermark embedding, subband processing and watermark detection.

6.3.1 Filters selection

We know that wavelet filters are able to decompose an image into different frequency bands and synthesize them back to an image exactly the same as the original one. More precisely, the original image data \( \{x_i\} \) can be decomposed into subbands \( \{c_i\} \) and \( \{d_i\} \) which can then be synthesized back to the original image \( \{x_i\} \) using (6.1) and (6.2). These transforms remain valid if \( x_i \) is made \( 2N \)-periodic, implying \( c_i \) and \( d_i \) being also \( N \)-periodic. This process can be conducted recursively to derive a hierarchy of subbands, as shown in Figure 6.3 for the Lena image (Appendix C). This remarkable feature makes it possible to perform various operations on the frequency bands of different resolution levels.
In this work we embed the watermark into the *selection* of wavelet filters of different classes such as orthogonal and bi-orthogonal filters in the process of decomposition. The watermark bits are translated into which filter is to be used for which level of image decomposition. The collection of such filters becomes crucial in this watermarking scheme. Suppose there is a pool of wavelet filters from different classes available for the collection, we will first derive a set of proper filters from this pool for a specific watermarking system. The criteria for the filter selection are to make the resulting decomposed subbands distinguishable enough from those by the other filters in the candidate filter collection. As a result a different ordering of the filters selected from this collection will be able to let the frequency bands be distinguished from each other.
Given a set of filters \( \{f_k\}, k=1,\ldots, n \), we first choose \( m \leq n \) ‘good’ filters \( \{\sigma_k\} \) so that the total difference \( \varepsilon_{ij}=\sum u \left| B^i_v-B^j_v \right| > \omega \) holds for \( i, j=1, \ldots, m, j \neq i \), where \( B^i_v \) denotes the frequency band at the final decomposition level using filter \( \sigma_i \) and \( u \) is the total number of bands at that level. The threshold \( \omega \) needs to be a proper value.

We tabulate in Table 6.1 all such \( \varepsilon \)’s for our 16 pool filters. Since the table is symmetric, the table entries below the diagonal are left as blanks. Then we select from \( \{f_k\} \) a subset \( \{\sigma_k\}_{k=1,\ldots, m} \), termed candidate filters, so that it has a maximum number of filters satisfying \( \varepsilon_{ij}>\omega \) for \( j \neq i \). In Table 6.1, where \( \omega=6 \times 10^5 \) we, for notational simplicity, denote the filters Daub4, Daub6, Daub8, Antonini, Villa1810, Brislawn, Brislawn2, Villa1, Villa2, Villa3, Villa4, Villa5, Villa6, Odegard, Mna2, Mna3 as \( f_1, f_2, \ldots, f_{16} \) respectively, and the fact \( \varepsilon_{ij}>\omega \) is marked by *.

These filters are well established elsewhere (see Appendix B for details). We now select the candidate filters bottom up from Table 6.1. In each step the filter set must consist of the filters from the last step. Finally we obtain the filter set as \( \{\sigma_k\} = \{f_{15}, f_{14}, f_{12}, f_{10}, f_7, f_5, f_3\} \) for \( m=7 \). We note that this set has the maximum number of distinguishable filters, rather than the other sets such as \( \{\sigma_k\} = \{f_{15}, f_{14}, f_{11}, f_9, f_6, f_4\} \) for \( m=6 \).

Obviously the selection of filters depends on the filter threshold \( \omega \) which reflects an acceptable distinguishing degree for the final subbands. This \( \omega \) may be determined by the individual watermarking algorithm. We will address this issue again later in the section.

**Table 6.1. FINDING CANDIDATE FILTERS**

<table>
<thead>
<tr>
<th></th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_4 )</th>
<th>( f_5 )</th>
<th>( f_6 )</th>
<th>( f_7 )</th>
<th>( f_8 )</th>
<th>( f_9 )</th>
<th>( f_{10} )</th>
<th>( f_{11} )</th>
<th>( f_{12} )</th>
<th>( f_{13} )</th>
<th>( f_{14} )</th>
<th>( f_{15} )</th>
<th>( f_{16} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
6.3.2 Embedding watermark

The image watermarking can be performed in sub-images. The original image may first be partitioned into image blocks and one or more sub-images may be decomposed independently into subbands and inserted with watermarks individually during the process. Then after synthesis of the subbands the sub-images are put together to form the watermarked image of the original size. An obvious benefit from this trivial step is that it provides additional security protection for the watermark as the image partition is kept unknown to the attackers. Another benefit is that it may facilitate special area protection, such as an object of an interesting area, since the specified watermark spreads only over the object region and detection of watermark would not be affected by the cropped background.

If an image is decomposed into $L$ levels, the original image being at level $L=0$, then there would be $4^L$ frequency bands in the level $L$, and there would be a maximum $3 \times 4^L$ analysis filters needed for ($L-1$)th level decomposition. The total number of filters to level $L$ is $4^{L+1}-1$, which are selected from the candidate filters $\{\sigma_k\}_{k=1,\ldots,m}$ from the last section. In other words, if we denote the decomposition
filter set as \( \{ F_k \}_{k=1,...,s} \), the set will consist of any times of \( \sigma_k \) in any order, which
leads to the unique final subbands. This filter set can be encoded to hold watermark bits. For the set of \( m \) filters, each single subband at level \( L \) is uniquely determined by its \( 2L \) path filters totaling \( m^{2L} \) different combinations. If \( K \) such subbands are used then, depending on the number of the initial overlapping sub-path, the number of different filter combinations can range from \( m^{2L} \) to \( K m^{2L} \). If we allow any single subband to be equally usable for encoding the watermark, then there are \( (4m)^L \) different bit patterns to represent (part of) the watermark. Hence such a single subband can store watermark of \( \log_2 (4m)^L = L[ 2 + \log_2 m ] \) bits, and more bits can be stored if one utilizes more such subbands.

There can be different ways for the watermark bits to be translated into the choice of decomposition filters and the decomposition path that leads to the subband that is to be partially sorted to flag the watermark existence. Here we describe such a simple translation scheme. Suppose that the watermark bits together form a watermark number \( U_0 \). Then we can repeatedly compute for \( i=0,1,...,L-1 \)

\[
U_i \equiv s_i \pmod{2}, \quad U_i = 2 \times V_i + s_i, \\
V_i \equiv S_i \pmod{m}, \quad V_i = m \times W_i + S_i, \\
W_i \equiv t_i \pmod{2}, \quad W_i = 2 \times X_i + t_i, \\
X_i \equiv T_i \pmod{m}, \quad X_i = m \times U_{i+1} + T_i.
\]

The \( s_i \) corresponds to horizontal decomposition with 1 standing for the low frequency band and 0 for the high frequency band, while \( t_i \) corresponds to the vertical decomposition with 0 standing for the low frequency band and 1 for the high frequency band. Since \( 0 \leq S_i, T_i < m \), they correspond sequentially to one of the
$m$ candidate filters. For instance, $S_i = 0$ corresponds to filter $\sigma_1$, $S_i = 1$ corresponds to filter $\sigma_2$, and so on. Of course the watermark number $U_0$ should not exceed the encoding capacity which means one should always have $U_L = 0$. Since the filters and decomposition path are determined by the watermark information the watermark can be chosen and modified to enhance the imperceptibility. This means the final subband targeted for the sorting process should avoid the use of the very smooth area (lowest frequency band) as well as the very high frequency band.

After a filter set is determined by the watermark bits, the image can be decomposed into frequency bands. We select certain specific bands for processing. Since the highest sensitivity of the human visual system is located at the lowest part of the frequency domain, and higher frequency band is more stable to human eyes but easier to be affected by noises, a compromise is to choose the middle bands for processing. We thus first select a frequency band, and then select coefficients in this band following a certain selection strategy. For example, we may take one from every 3 coefficients in sequence from the selected band. Next we sort this set of coefficients and put them back sequentially in the locations of all those selected coefficients. The synthesis process with the modified subbands generates a new image slightly different from the original one. This watermarked image that looks not much different from original image actually contains the watermark information, identifiable by the unique filters and paths to the specified subbands.

### 6.3.3 Detection

The watermark detection is carried out by examining the deviation of coefficients’ sorting trend in the chosen bands with the correct filters. Since the
frequency subbands are uniquely determined by the filter set, the sorting property should remain consistent within admissible noises. The locations at which the coefficients are to be extracted for sorting, i.e., the selection strategy of the locations for sorting, can be image dependent. We here aim to find a monotonically non-decreasing curve that minimizes the error between this curve and the coefficients. If we denote the coefficients by \( z(x) \), indexed by \( x \), and the curve by \( \Gamma(x) \), then the approximation error \( \delta \) is set to

\[
\delta = \| z(x) - \Gamma(x) \| = \frac{1}{n} \sum_{x=1}^{n} |z(x) - \Gamma(x)|,
\]

where \( n \) is the number of selected coefficients. To approximate the solution of this problem, we initially relax the monotonic condition so that we can make use of the least squares method to approximate the curve, \( P(x) \) (see an example in Figure 6.4). We, in fact, will find a 3\(^{rd}\) order polynomial \( P(x) = \sum_{k=0,\ldots,3} a_k x^k \) for the curve so that \( \psi(a_0, a_1, a_2, a_3) = \sum_{x=1,\ldots, n} (P(x) - z(x))^2 \) is minimized. We now introduce an additional parameter \( \xi \) to compensate the curve deviation from the monotonicity. For this purpose, we partition the curve \( P(x) \) into monotonic segments via solving \( P'(x) = 0 \). The solution will be:

- If there are complex roots, then \( \xi = 0 \).
- If there are two repeated roots, then for \( P''(x_0) > 0 \) a minimum value exists, and we set \( \xi = \sum_{x=1,\ldots, x_0} |P(x)| \); if \( P''(x_0) < 0 \) a maximum value exists, and we set \( \xi = \sum_{x=x_0,\ldots, n} |P(x)| \).
• If there are two distinct roots \(x_0\) and \(x_h\) with \(x_0 < x_h\), for \(P(x_h) > P(x_0)\) we set

\[
\xi = \sum_{x=x_0, \ldots, x_h} |P(x) - P(x_0)| + \sum_{x=x_h, \ldots, n} |P(x) - P(x_h)|; \text{ for } P(x_h) < P(x_0) \text{ we set }
\]

\[
\xi = \sum_{x=x_0, \ldots, x_h} |P(x) - P(x_h)|.
\]

![Figure 6.4 Approximate curve](image)

As shown in Figure 6.5, the monononicity deviation is measured by the shaded non-increasing areas. The combined overall error is thus proposed to be \(\delta' + \lambda \xi\), where \(\delta' = ||z(x) - P(x)||\) or its approximation and \(\lambda \geq 0\) is a coupling constant.

The detection can be carried out regardless of the availability of original (noise-free) watermarked image. If the original watermarked image is available the approximated sorting curve \(Q(x)\) can be obtained by the correct filter set. Since the approximated curve \(P(x)\) for the examined image is obtained by the same filter set, we can determine if the watermark exists by checking if \(\delta' = ||Q(x) - P(x)||\)
Figure 6.5 Deviation from monotonicity

Figure 6.6 (a) Subband from filter Daul4. (b) Subband from filter mna3.

$P(x)\|<\tau$. The detection threshold $\tau$ is quantified to accommodate certain noise distortions and will be discussed in the next section. If the original watermarked image is unavailable the measurement can be based on the deviation from the approximation curve itself in the examined image, which is $\delta'+\lambda\xi$. The reason we introduce the $\xi$ here is that $\delta'$ itself can not represent the ascendant sorting information. For example, a decreasing curve resulting from a specific set of filters could also lead to a small $\delta'$ and mislead to the watermark presence if $\xi$ is not introduced. This is shown in an example in Figure 6.6 where the subband comes from filter Daul4 and mna3 (Appendix B) respectively at level 4. The watermarking and detection steps are listed in Figure 6.7.
6.3.4 Adjustment for watermark requirements

6.3.4.1 Invisibility

We have mentioned about selecting a partial set of coefficients from the subbands in section 6.3.2. The aim to select some coefficients, i.e. elements, from the frequency subband rather than use the whole subband is to achieve better watermark invisibility and also to make it harder to reveal to attackers so that it provides better security. As mentioned before, the coefficients can be selected each every 3 coefficients in sequence. The selection strategy actually can be user defined, for instance by using the different subset of coefficients and its ordering to ‘span’ the subband. Any particular walk can be used to pick the coefficients (see 4 different walks depicted in Figure 6.8). The visual quality of synthesized image will be affected by the number of the selected coefficients, band size and the position of bands. More coefficients and larger band size would decrease visually the image quality. However fewer coefficients may result in a less stable sorting curve, and thus less robustness to noises. We illustrate the effect of the number of coefficients by an example in Table 6.2, where images Lena, Goldhill, Peppers, Baboon, Barbara, Cameraman (Appendix C) are denoted by A, B, C, D,
Watermarking

1. Find suitable and distinct filters \( \{\sigma_i\} \).
2. Determine the set of filters \( \{F_i\} \) for the subband decomposition for the specified image.
3. Select or design a coding method for embedding the watermark bits in the filters and the path.
4. Decompose the image with the selected filter set.
5. Select frequency subband/s for processing.
6. Select or design a coefficient selection strategy, and extract a subset of coefficients accordingly.
7. Sort the coefficients and place them back in the band based on certain specifically chosen rules.
8. Synthesize the frequency bands into a full image.
9. Determine the detection threshold \( \tau \).

Detection

1. Decompose the image with correct filters \( \{F_i\} \).
2. Recover the selected frequency subband/s.
3. Extract coefficients in the original sorting order.
4. Find the approximate curve \( P(x) \) for sorting coefficients.
5. Calculate the coefficients deviation \( \delta' \) from the curve \( P(x) \) and monotonicity deviation \( \xi \).
6. If \( \delta' + \lambda \xi < \tau \), then the watermark exists, otherwise, watermark does not exist, or it can’t be determined.

Figure 6.7 Watermarking and detection scheme
E and F respectively. The sizes of image A, D and a frame G from a video are 320x240 and the other images have size 512x512. We also note that ‘partial’ in Table 6.2 means the percentage of the coefficients in the band, and ‘bands’ there means the number of bands involved in the modification.

![Figure 6.8 4 walks](image)

| Image | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | B | C | D | E | F | G |
| Partial | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 | 1/2 | 1/3 | 1/4 |
| Level | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| Bands | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Psnr | 50 | 51 | 52 | 48 | 50 | 51 | 45 | 46 | 47 | 43 | 45 | 47 | 61 | 63 | 63 | 58 | 59 | 59 | 57 | 55 | 54 | 50 | 60 | 62 | 48 | 65 | 55 | 54 |

The band size in level 3 is 32x32 for image A. From Table 6.2 we know when band size is smaller it allows more bands to be involved in modification at the same time without perceptually degrading the image quality. The other images indicate similar results.
We note that how the sorted coefficients are put back can also be adjusted to reduce the watermark perceptibility in the image as well as in the modified bands. For example, if the sorted coefficients are put back corresponding to the order of mean value of each slot’s neighboring coefficients, rather than the sequence order when coefficients are extracted, then the modified band is even more similar to the original one.

6.3.4.2 Noise resistance

The detection threshold $\tau$ determines how much the deviation of coefficients from the detected curve is allowed to assert the watermark existence. The deviation is not only caused by the different filters (including the ordering of the filters), but also caused by the noises. It is, however, hard to determine whether the detected deviation comes from the noises or the incorrect filters, e.g. if the noise is large enough, the deviation would then be as large as using incorrect filters. This threshold needs to exclude the incorrect filters, and also be able to resist certain amount of noises. Our experiments show that our scheme can resist white noise up to at least 4%, and that the threshold $\tau$ may depend on the filter set in Table 6.1.

6.3.4.3 Filter threshold

The filter threshold $\omega$ is used to choose the filters so that the resulting subbands are sufficiently distinguishable under different filtering sequences. The larger the threshold, the more distinguishable the resulting subbands. In Table 6.1, the filter threshold is set to $6 \times 10^5$, resulting in the chosen filters leading to the average
difference of 9.15 per coefficient. This threshold \( \omega \) will also impact on the detection threshold \( \tau \).

### 6.3.5 Hybrid models

It is possible to utilize the orthogonal wavelets to greatly expand the wavelet filters collection, so as to significantly enlarge the value \( m \) to allow more watermark bits to be stored. If we define \( A(z) = \sum_{k \in \mathbb{Z}} A_k z^{-k} \) with \( A_k \) being a 2x2 matrix of \( (h_{2k}, h_{2k+1}) \) in the 1st row and \( (h_{1-2k}, -h_{-2k}) \) in the 2nd, then \( A(z) \) induces orthogonal wavelet filters iff \( A(z) \) admits the factorization as in (6.3). The free parameter \( \theta \)'s can be used to carry the watermark bit patterns. The \( \theta \) parameter seems homogenous and linear at differentiating the wavelet filters, and the orthogonal filters tend to be in a different filter league. The details of this parametric watermarking will be briefly given in the next section as an illustration.

Other bi-orthogonal filters with continuous free parameters may play a similar role but the scaling may not be linear anymore. For an image represented by \( \{x_k\}_{k \in \mathbb{Z}} \), suppose the \( \{h_k\} \) and \( \{\overline{h}_k\} \) satisfy the bi-orthogonality (6.4). The filter coefficients \( \{h_k\} \) and \( \{\overline{h}_k\} \) will generate subbands \( \{c_j\} \) and \( \{d_j\} \) via (6.1), and the subbands can be synthesized by the synthesis filter there. The bi-orthogonal filter can also be generated with free parameters (Jiang and Guo 2003), e.g. from (6.5), and this can provide many filters to choose from. However, extensive investigation with such hybrid models is yet to be conducted.
6.3.6 Analysis and experiment

In the following, we illustrate the algorithm with the standard test image lena.jpg (Appendix C). The filters selected are as in Table 6.1; they are \{σ_i\}={mna3, mna2, Villa5, Villa3, Brislawn2, Villa5, Daub8}. Let the filter set for decomposition be \{F_i\}={σ_7, σ_7, σ_1, ..., σ_1, σ_5, ..., σ_5, σ_5 ..., σ_5}, implying the same filter is used for each level of decomposition. The filter path for any subband becomes \{σ_7, σ_1, σ_5, σ_5\}. The sorting coefficients were selected each for every three coefficients sequentially. After sorting they were put back in the selected band. The subband before modification is shown in Figure 6.9 (a) while the modified subband with the sorted coefficients which were put back in the sequence is shown in Figure 6.9 (b), looking similar.

![Figure 6.9](image)

To assert the watermark existence, the coefficients were extracted and the sorting information was examined. The results are illustrated in Figure 6.10.

In Figure 6.10 (a), the blue (dot) line depicts the sorted coefficients with the correct filter set, the red (solid) line is the sorted coefficients with 4% white noise, and the green (dash) line displays the sorted coefficients obtained from
Figure 6.10 (a) Sort coefficients. (b) A local zoom out of (a). (c) Approximation curves.

another different filtering path \{\sigma_1, \sigma_5, \sigma_7, \sigma_7\}. A local region in Figure 6.10 (a) is magnified in Figure 6.10 (b). Figure 6.10 (c) shows the approximated curves using least squares method. The deviations of coefficients from the blue, red and green curves are calculated as 1.09, 4.21 and 30.56 respectively. If using the original watermarked image for detection, the deviation of coefficients (red line in (a)) from the blue curve in (c) is 4.24, and the deviation of the coefficients (green line in (a)) from the blue curve is 30.62. We can see that it is not much different whether or not the original watermarked image is used for detection. Similar results are observed for the other test video frames and images as illustrated in Table 6.3. We note that
all filtering paths for the table are distinctive and uniquely identify the subbands, and the watermark is extracted by the translation of the actual filter path.

<table>
<thead>
<tr>
<th>Error</th>
<th>penguin</th>
<th>parrot</th>
<th>meerkat</th>
<th>car</th>
<th>goldhill</th>
<th>peppers</th>
<th>baboon</th>
<th>barbara</th>
<th>camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct filter</td>
<td>1.37</td>
<td>1.83</td>
<td>2.94</td>
<td>2.03</td>
<td>1.7</td>
<td>1.28</td>
<td>5.05</td>
<td>0.86</td>
<td>4.22</td>
</tr>
<tr>
<td>Different filter</td>
<td>19.16</td>
<td>19.57</td>
<td>26.63</td>
<td>19.16</td>
<td>16.02</td>
<td>16.67</td>
<td>28.72</td>
<td>16.63</td>
<td>27.01</td>
</tr>
</tbody>
</table>

### 6.4. Watermarking via filter parameters

Many orthogonal filters exhibit good performances in image decomposition and synthesis and can be generated via a set of free parameters. Watermarking proposed in the previous section can be applied to the orthogonal filters utilizing the filter parameters to carry the watermark bits. In the following we describe briefly how the orthogonal filters can be generated by the free parameters and the application to our proposed watermarking mechanism.

#### 6.4.1 Wavelet filter selection by the parameters

We know the orthogonal filters can be generated by the \( \theta \)'s in (6.3). Hence these \( \theta \)'s may be used to directly carry watermark bits as they correspond to the choice of the wavelet filters. For instance, when \( q = 2 \), \( A(z) \) in (6.3) is expressed as

\[
A(z) = \begin{bmatrix}
  h_0 - z^{-1} h_2 - z^{-2} h_4 & h_1 - z^{-1} h_3 + z^{-2} h_5 \\
  -\partial g_0 - \partial z^{-1} g_2 - \partial z^{-2} g_4 & -\partial g_1 - \partial z^{-1} g_3 + \partial z^{-2} g_5
\end{bmatrix},
\]

the induced filters are thus represented by
\[ h_0 = \cos(\theta_0) \cos(\theta_1) \cos(\theta_2), \]
\[ h_1 = \cos(\theta_0) \cos(\theta_1) \sin(\theta_2) \]
\[ h_2 = -\sin(\theta_0) \sin(\theta_1) \cos(\theta_2) - \cos(\theta_0) \sin(\theta_1) \sin(\theta_2), \]
\[ h_3 = -\sin(\theta_0) \sin(\theta_1) \sin(\theta_2) + \cos(\theta_0) \sin(\theta_1) \cos(\theta_2), \]
\[ h_4 = -\sin(\theta_0) \cos(\theta_1) \sin(\theta_2), \]
\[ h_5 = \sin(\theta_0) \cos(\theta_1) \cos(\theta_2), \]
\[ g_0 = -\vartheta h_5, \quad g_1 = -\vartheta h_3, \quad g_2 = -\vartheta h_2, \]
\[ g_3 = \vartheta h_2, \quad g_4 = -\vartheta h_1, \quad g_5 = \vartheta h_0, \]

where \( \theta = \pi/4 - \theta_0 - \theta_1 \), and \( \theta_0 \) and \( \theta_1 \) are two free parameters. We can generate a pool of candidate filters by varying these free parameters, and then follow the general procedure outlined in the previous section to do the watermarking. However if this pool somehow automatically gives rise to sufficiently distinguishing filters due largely to the homogeneity of the \( \theta \)'s, then the watermarking process can be simplified and the watermark bits can be directly embedded into the choice of those free \( \theta \) parameters.

If the watermark bits are represented by \( \{w_j\} \) with each \( w_j \) being either 0 or 1, then they can be directly embedded into \( \theta \) via, for example,

\[ \theta = \nu (\sum_j w_j 2^j) \Delta \theta, \]

where \( \Delta \theta \) is a chosen \( \theta \) step size and \( \nu \) is a balancing constant. The main criterion for selecting the step size \( \Delta \theta \) is that the error caused by the difference of the original watermark and that extracted through an incorrect filter should exceed clearly the difference caused by the tolerable white noises. Thus the choice of \( \Delta \theta \) is a balance
between better robustness and higher storage capacity for the watermark. For example, a larger $\Delta \theta$ gives a more robust watermark at the cost of storing fewer watermark bits per filter. Hence $\Delta \theta$ can be determined by analyzing the effect of white noises as well as the distinguishing power on the induced subbands. A predefined image pattern can be used to replace the selected subband so as to distinguish the watermarked image and original one. In this sense, the sorted band used in the last section is a special case of such patterns. To improve the watermark resistance to the white noises, a particular pattern, or behavior indicative of the watermark existence, should be made detectable by using the $\theta$ that carries the watermark as long as the noises added to the image do not cause visible visual degradation. However, if the $\theta$ changes even only one step in its watermark bits, such a pattern will not be detected by using the filters generated from this $\theta$. Hence, if we want the system to be more robust to noises we can artificially choose a higher threshold for $\Delta \theta$. Moreover, the $\theta$‘s can not only contain a predefined watermark, but also contain a private encryption key or can be optimized to improve the quality of filters such as by imposing a maximum number of vanishing moments.

6.4.2 Embedding and detection

Although essentially just a special case of the general scheme proposed in section 6.3, the use of parametric orthogonal wavelet filters allows the embedding and detection process to be addressed in more direct or different manners. We need to determine the $\Delta \theta$ and calculate the $\theta$ according to a given watermark $W$, and then derive the wavelet filters $f_w$ by (6.3) or (6.8). Next we use the watermarked filters $f_w$ to decompose the image through the chosen path to reach the selected subband.
After processing the selected subband by such as substituting it with a predefined pattern, all the subbands are synthesized together to the watermarked image in full size. The watermark here is not exposed directly to the users along with the image, and it can only be proved via the existence of the pattern through the correct watermarked filters, thus improving the robustness and security of the watermark. The extra free $\theta$'s may also be used to optimize the filters and, subsequently, the watermarking performance.

The watermark detection process does not need the original image. It is based on extracting the selected subband with the correct filters. With the pattern and the optional private key provided by the owner of the image, one can calculate the RMS between the designed pattern and the selected band of decomposition obtained from watermarked filters and the proper decomposition path. If the RMS is less than the detection threshold, then it can be asserted that the predefined watermark does exist on the image. If the RMS is larger than the detection threshold, it can not immediately be asserted that no watermark exists since this error may be caused by large noises as we have determined the detection threshold with the capacity of noise tolerance to a certain level. If the noise exceeds this level, the RMS would exceed the detection threshold.

The detection process will require watermarked filters, pattern, private key and/or other information such as the decomposition path. One can not extract the right pattern from the image with only partial information. Also, the pattern can be further scrambled to make it difficult to guess for the content or location. In addition, a private key may help prevent others from correctly interpreting the
watermark bits in watermarked $\theta$'s. To summarize, the watermarking system of parametric wavelet filters is a special alternative instance of the general methodology outlined in the previous sections, and achieves good robustness, security and invisibility for the watermark.

6.5. Image Synchronization and the Detection Algorithm

If a digital image is printed out and then scanned to generate another digital image with the same content, can the watermark survive and still be extracted? In this section, we investigate such watermark detection in a different digital image with the same content, e.g., from the rescanned copy of a pirated image.

6.5.1 Synchronization for rescaled images

We first approach this problem for a simplified case in which the reproduced image has the same resolution as the original. If a digital image has been rotated and rescaled, we will first try to adjust it back to match position-wise with the original image. If we can determine two points in the rotated image whose corresponding positions are known in the original image, then we can rotate the image back to its original position. As we know, many pixels may have similar contribution to the appearance of an image but an image region characterized by a set of local pixels is more likely to be unique in the same image. Should the uniqueness fail, one may still choose a different set of such local pixels for the purpose. Hence we can search a corresponding pixel position in the rotated image in terms of pixel set of a characteristic feature.
The method we propose to adjust the rotated image needs the original watermarked image, which is supposed to be available in the detection process for such cases. We choose our pixel sets by circle since it is rotation invariant, and the pixels included in the circles are very close to the same set of pixels in either the original image or a rotated image since our measurement will be independent of the rotation angles. By increasing the radius we can obtain several sets of pixels for a specified centre, and we denote by \( r \) the radius of the circle centered at pixel \( s \).

For each of the concentric circles, let \( I_i \) for \( i=1, \ldots, n \) represent the intensities of all the pixels there, and we define the average \( \delta_1 \) and the other average powers \( \delta_{k+1} \) by

\[
\delta_1 = \frac{\sum_{i=1}^{n} I_i}{n}, \quad \delta_{k+1} = \left( \frac{\sum_{i=1}^{n} |I_i - \delta_1|^k}{n} \right)^{1/k},
\]

(6.9)

where \( k = 1, 2, 3, 4, \ldots \). We compare these feature values from the original watermarked image with those obtained from the rotated image in the same way. The feature values of the corresponding pixel in a rotated image should yield the minimum error when compared with those for other pixels. However, the error caused purely by rotation operations need to be taken into account as well. This is because the pixels may have been interpolated during the rotations and a rotation forward and then backward to the original position may cause a slight change of grey values. To minimize this effect, we will first calculate the difference between pixels and average intensity of the corresponding concentric circles, and then increase the power of these differences via (6.9), to match another set of feature values. These feature values will thus better characterize the local region of a pixel.
point. The final total error $\varepsilon$ between the two local regions corresponding to the original image and the rotated image respectively is then measured by

$$
\varepsilon = \sqrt{\sum_{k=1}^{m} (\delta_k^r - \delta_k^o)^2},
$$

(6.10)

where $m$ is the number of concentric circles to be used, and the super indices $r$ and $o$ as in $\delta_k^r$ and $\delta_k^o$ indicate that they are the $\delta_k$ defined in (6.9) for the reproduced image and the original image respectively. In other words, the $\varepsilon$ represents the total error of feature values between the two images. We consequently stipulate that the matching pixel points in the two different images should result in minimum errors.

If an image $I$ is rescaled in terms of its intensity into image $y$ via a linear transform $y = a \ I + b$, we use the least squares approach to match the average intensities transformed from the rescaled image with the original values by minimizing the error $\varepsilon$ defined in (6.10). More precisely, if we denote by $\delta_k^a$ the approximated average intensities of the transformed image, then $\delta_k^a$ will also be linearly related to the average intensity $\delta_k^o$ for the original watermarked image, i.e.

$$
\delta_k^a = \alpha \delta_k^o + \beta,
$$

(6.11)

The task thus becomes to find out $\alpha$ and $\beta$ so that the difference between $\delta_k^a$ and $\delta_k^o$ is minimum in terms of least squares. This means the $\alpha$ and $\beta$ can be obtained from the following linear equations
\begin{equation}
\begin{bmatrix}
\sum_{k=1}^{m} (\delta_k^r)^2 \\
\sum_{k=1}^{m} \delta_k^r
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} =
\begin{bmatrix}
\sum_{k=1}^{m} \delta_k^a \delta_k^r \\
\sum_{k=1}^{m} \delta_k^a
\end{bmatrix},
\end{equation}

Once \( \alpha \) and \( \beta \) are obtained, we can calculate \( \delta_k^a \) via (6.11) and evaluate the error \( \varepsilon \) defined by (6.10) with the \( \delta_k^r \) there substituted by \( \delta_k^a \). The corresponding pixel position we want to obtain should thus be the one with minimum error \( \varepsilon \) to the selected pixel position in the original watermarked image, or the one very close to the minimum value due to the subtle overhead errors of cascading rotations. Once we obtain a pair of corresponding pixel positions in the reproduced image, we can calculate the rotation angle, if any, and then use the weighted distance to get the new intensity by

\[ b_i = \sum_{j \text{ over } 4 \text{ neighbors}} a_j \frac{a_j}{\left(d_{ij}^2 + d_{ij}^2\right)^{5/2}}, \]

where \( d_{ij} \) and \( d_{ij} \) are distances from the neighbour pixel to the selected pixel in horizontal and vertical directions respectively, see Figure 6.11. The figure shows that if we rotate image \( A = (a_i) \) into image \( B = (b_j) \), then a typical pixel point \( s_i \) in \( B \) has four neighbour points represented by the indices \( j \) to \( j+3 \), and these four neighbour points are mathematically mapped from the pixels in the original image \( A \). This calculation is largely necessitated by the fact that the pixel points in \( A \) may not fall exactly on the pixel grid of image \( B \) after the rotation. Hence we calculate the intensity of selected pixel via (6.1). We note that the exponential \( 5/2 \) is derived empirically.
An alternative approach to deal with a linearly rescaled image is to remove the effect of the unknown \( a \) and \( b \) in the perceived image transformation \( y_j = ax_j + b \), so that the searching of the pixel positions is carried out in a transformed space that does not involve \( a \) and \( b \). One way to achieve this is to first eliminate \( b \) in \( y \) via \( e_j = y_j - y_j \), and then eliminate \( a \) via \( \bar{a} = e_j / e_j \), where \( y_j \) and \( e_j \) represent the intensity average of the pixels inside a circle of a given radius centered at the pixel position \( y_j \) and \( e_j \) respectively. Obviously the resulting image \( \bar{a} \) is well defined anywhere not at the centre of a circle of constant pixel values. When this process is applied to both the reproduced image and the original watermarked image, then the pixel position matching can be done via (6.10) directly rather than via (6.11) and (6.12). The limitation of this straightforward approach is that it may not work well in the smooth or constant area of an image due to the close to zero denominators, and thus would need additional adaptation to such circumstances.
We note that if a rescanned or reproduced image is obtained in a different resolution, then certain quantization and interpolation of pixel values may have to be performed in order to adjust the resolution to that of the original watermarked image. Also, the energy of the reproduced image needs to be adjusted to the same level as that of the original watermarked image.

### 6.5.2 Detection procedure

We apply the synchronization techniques developed in the above subsection to detect the watermark in a rescanned image. Since a rescanned image would, in general, result in rescaled brightness, minor rotation and different resolution due to the scanning process, the methods we proposed above are used to restore the image to its original format. Then the detection process can be carried out using the threshold or patching method to detect the watermark in the restored image.

The patching method we developed (Huang and Jiang 2003) for the watermark detection substitutes each block of watermarked image with the corresponding block in the suspected image to create a patched image, and then calculates the RMS of the resulting selected pattern via the correct parameterized wavelet filters. When the watermarked image has larger noise than that in the suspected image, if the suspected image contains specified watermark, the RMS should decrease when the patching area increases.

The scanning process inevitably produces distortion on the reproduced image. If we add noise larger enough on the original watermarked image so that it overcomes the effect of distortion on the reproduced image, the patching method should perform well in the reproduced image watermarking detection. The detection
process can be summarized as the following:

1. Adjust the resolution of the reproduced image $I_s$ to the same resolution as the original watermarked image $I_w$.

2. If the reproduced image $I_s$ has undergone a certain rotation, rotate it back to the right position as the original watermarked image $I_w$ using the above method.

3. Rescale the energy of the reproduced image $S$ to the same brightness as the original watermarked image $I_w$.

4. Apply the patching method on the reproduced image $I_s$ after the above adjustment. Inject large noises to the original watermarked image $I_w$. Crop the reproduced image into several pieces and patch them back to the original watermarked image $I_w$ at the corresponding positions. Use the predefined wavelet filters to filter the patched image, and then obtain a group of RMS values of filtering band and pre-selected pattern.

5. If RMS values decrease with the increasing of pieces of reproduced image, it can be asserted that the watermark is present in reproduced image. In other words, watermark survives the rescanning process.

The noise added to the original watermarked image has to be determined since different scanning process may result in different quality of reproduced image. A simple alternative for this is to make use of the original image $I$ rather than the watermarked image $I_w$ in the patching method.

The watermarked image may undergo certain distortion such as cropping and rescaling. A rescanned image can be treated as a cropped image. As we know, the
wavelet-based watermarking can spread the watermark all over the image. If one crops a part of the image, it may still contain sufficient information on the watermark. The strategy we will use here is to derive the watermark from the rescanned image with the help of the original watermarked image: considering the original watermarked image with the rescanned and/or cropped image replacing its corresponding area. To this end, we first add the noises $M$ to the full-sized watermarked image $I_w$, this noise can mollify the effect of noises $N$ in the cropped image if $M > N$. In other words, the difference between the patched image and the full-sized original watermarked image should decrease when the cropped area increases. If the cropped image has no specified watermark, such a difference should increase instead when the area of the cropped image becomes larger. This sharply contrasted trend of decreasing or increasing difference proves to be clear and universal, and suffices to differentiate cropped images corresponding to a given watermark.

6.5.3 Experiments

In the experiments for image Lina256.jpg, we first embedded into the image with the watermark signature ‘lena’, as in (Huang and Jiang 2003), resulting in an original watermarked image $I_w$. Then we rotated the watermarked image by a certain angle as shown in Figure 6.12 (a). We choose the pixels that have significant intensity differences from the neighbour pixels in the original watermarked image for our test purposes. There are a total of 58 pixels whose difference of grey value is more than 70 to the left of and up its neighbour pixels. In Table 6.4, in the case of the rotated image without rescaling, 56 out of the 58 chosen pixel points in the
rotated image can be located. When we added up the terms of power $p$ from 1 to 2 in (6.9), the accuracy rate is 96.6%. If we added up those differences in (6.9) from power $k$ 1 to 3, it would achieve a similar result; the rate is much higher than those achieved by applying only one of the powers in the search. We can easily see that with the use of only one power many pixels may fail to be matched.

![Figure 6.12](image)

(a) Rotated watermarked image. (b) Reproduced image.

In the case of the rotated image having been linearly rescaled, the success rate also increases with the increase of the radius we chose for the circles when using a combination of powers, though the method does not work on the individual power. It works even better if we add up the terms corresponding to powers 1, 2 and 3. As shown in Table 6.4, 85% pixels have been found when the radius of circles varies from 11 pixels to 23 pixels with the step size of 3 pixels, shorthanded by 11:3:23. We note that the numbers of pixels found in the table has the tolerance of maximum four pixel distance.

Next, we printed out the original watermarked image and used a scanner to generate another digital image with certain rotation and different resolution as
shown in Figure 6.12 (b).

**Table 6.4** Successful number of pixels matched in the rotated image

<table>
<thead>
<tr>
<th>Powers Used</th>
<th>Rotated Image</th>
<th>Radius</th>
<th>1+2</th>
<th>1+2</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>In (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rescale</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No rescale</td>
<td>5:3:17</td>
<td>56</td>
<td>55</td>
<td>30</td>
<td>39</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Rescale</td>
<td>5:3:17</td>
<td>22</td>
<td>37</td>
<td>3</td>
<td>4</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>y = 1.5*I</td>
<td>8:3:20</td>
<td>36</td>
<td>45</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>+ 5</td>
<td>11:3:23</td>
<td>38</td>
<td>49</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

By applying the above methods we obtained the restored image. We then performed the watermark detection process on the restored image. Firstly we chose an area of 192x192 in the reproduced image, divided it into four equal pieces, and patched it back one by one to the original watermarked image that has been added with noise. We can obtain the selected band via predefined wavelet filters, and calculated its RMS. Another test was to patch the reproduced image without watermark to the original watermarked image. Figure 6.13 (a) and (b) illustrate the results. The RMS trend shows a strict decrease when the reproduced image contains the watermark while the noise ratio added to the original watermarked image is 0.2.

However, the result is not obvious when the noise added to the original watermarked image is larger than 0.2. It is also hard to predict the suitable amount of noise that should be added. We may use the original image instead of the original watermarked image for detection. Figure 6.13 (c) shows the RMS
Figure 6.13. (a) Reproduced image contains watermark. (b) Reproduced image contains no watermark. (c) Reproduced image contains watermark, using original image
trends decrease more sharply if the noise becomes larger. We finally note that scanning quality is of significant importance to the success of the watermark detection. This is of course natural: a low quality rescanned image will destroy the embedded watermark the same way it destroys the entire image itself.

6.6. Video watermarking

The watermarking mechanism for images using wavelet filters should be directly applicable to individual video frames. A particular consideration for video watermarking is the data rate of the video. We know a video contains a large volume of data, and the sheer data volume may cause an efficiency problem for watermarking. Hence low computational load is often a must for watermarking video frames. Furthermore, the video watermarking may be under particular attacks from frames changes due to different formats and the video operations such as dropping frames and averaging frames. On the other hand, the invisibility requirement could be less stringent in video watermarking since human eyes are not sensitive to the visible flaws on the fast moving frames.

To reduce the computational cost, one way is to select some typical frames relevant to the video format. Another is to watermark local regions. Although wavelet based watermarking generally spreads the watermark all over the whole image, making the watermark more robust, it can also be conducted on the sub-images as mentioned in 6.3.2. These sub-images could, in fact, be the areas of interest that need protection by watermarks. The sub-images as those in Figure 6.14 can also be selected for the watermarking independently and finally join together to form a full frame. Hence one can focus on an object area of interest in a frame, and
protect it with a watermark rather than for the full frame. This reduces the volume of data to be processed, and the size and the location of the region can be user defined. It is potentially possible to even embed into the watermark a certain object cue a tracking algorithm may make use of.

![Sub-images](image)

**Figure 6.14** Sub-images

If one is interested in watermarking all the video frames, although the above watermarking algorithm can till be applied the computational cost will be very high since all the frames are involved. One solution to overcome this problem is to apply the algorithm only on the temporal frames, where each column of pixels is the column of pixels on the same position of frames along the temporal. For instance, given a video of frame size 250x200 and of 500 frames in total, one can construct 250 frames of size 500x200. Each row of pixels in such a frame actually reflects the color changes of the same pixel in the original video frames. It is unnecessary to perform watermarking on the 250 frames since each such frame carries certain information of all the original frames. A few of such frames done for the watermarking may be sufficient to signify the watermark presence throughout the video sequence. However, more elaborate work on video watermarking with wavelet filters and on local regions of interest is beyond the scope of this thesis and will thus be pursued as our future work.
6.7. Summary

In this chapter, we presented a watermarking scheme where wavelet filters are selected and organized to carry the watermark information through the identification of the sorting information on the selected coefficients from the subbands targeted for the watermark existence. The filter path consists of filters from a variety of filter classes, such as orthogonal and/or bi-orthogonal filters, with or without free parameters. These wavelet filters are predefined and randomly chosen from the filter pool where filters are able to derive a sufficient distinguishing degree of subbands. The strategy of choosing the coefficients in the transform domain for the processing is user-defined and can provide further watermark security. The watermark is embedded into the choice of wavelet filters and the filtering path; rather than a pattern as in the traditional case. The watermark detection is carried out by examining the deviation of wavelet coefficients’ sorting trend in the chosen bands, with the correct filters being used in the watermark embedding process. The method is able to resist certain white noises and illumination and geometric distortions. Furthermore, the watermark can be embedded into the parameters that are used to generate wavelet filters such as the parametric orthogonal wavelet filters, and no original image is required for the detection process.

We also investigated the problem of image synchronization and its application to the detection of watermarks from the rescanned images. It is important to bear in mind that a reproduced image has undergone certain distortion from the original watermarked image which means the information hidden in the image is also weakened to a degree in the same process. Once a reproduced image has been
synchronized, we can utilize a modified patch method to detect the hidden information if the quality of rescanned image is high. The method for rotating and rescaling the reproduced image back so as to align with the orientation of the original image basically has achieved a high successful rate when suitable pivotal pixel positions are chosen. Rescanning is a process that, by its nature, introduces typically excessive noises and different scales of grey intensities. Based on the understanding that such changes are homogeneous because the scanned images will not twist themselves, linear changes may pull them back to be in line with the original images, and enable subsequent watermark detections to be performed. An outlook on watermarking video sequence is also briefly presented and explained.
7.

CONCLUSION

The object tracking in video sequences has been investigated in this thesis through feature selection and modeling, Bayesian estimation as well as shape refinement and consolidation methodologies. The significant object features in a scene have been modeled from different perspectives in terms of distinguishing, stable or dominant features, particularly in colors. The color distribution density has also been integrated implicitly or explicitly with the spatial relationship among the visual pixels to improve the accuracy of object capturing in varied tracking conditions with deformable object shape, illumination variation, local appearance change or partial object occlusion. A lower computational cost for locating the object of interest in a new frame is further achieved through statistics but with fewer samples of the object’s appearance in the method of dominant elements.
We developed the object appearance models based on the color distribution through the kernel density estimation. These models of statistical nature have been designed to reduce the impact of the undesirable distribution due to possible illumination or local appearance changes, thus improving the stable performance of the model throughout the video sequence. The models may require fewer samples to characterize the object for the tracking purpose, and can also be established in different ways to suit the characteristics of an individual object. The model of annular regions, for instance, works better for the object of unstable contour while the model of sparse samples is more suitable for the object with sizable change in local appearance. The computational cost is reduced not only by the small number of statistical samples instead of a large amount of raw pixels for the object appearance, but also by the choice of the significant samples in the searching step in some particular cases. Such models are less affected by the shape change of the object.

The color differentiation between the object and background is an effective cue for tracking an object in a moving background. To better extract such a characteristic, we introduced the concept of local background to exclude the impertinent visual noises. We developed a scheme to make use of larger density differences in color within different color spaces, representing the prominent feature contrast in color or texture for the object and the background. The object shape in a newer frame is subsequently traced out according to the probability of a pixel being on the object after applying a Bayesian estimation or matching directly the dominant elements. The dominant elements are designed to
characterize the distinctiveness of colors in the object vis-à-vis the background, and they are dynamically updated from the previous frame to the current one. The direct matching of the dominant elements has successfully removed the need to iteratively search the object location which is commonly found in the other tracking algorithms. These proposed approaches are effective at capturing the object in the subsequent frames, and they can survive a good deal of non-rigid object movement, moving background, partial object occlusion, illuminate variation and internal change of the object appearance.

We also investigated the tracking of a camouflaged object in a nature scene. We proposed the weighted regions consolidation method to synchronize the frames geometrically to reduce the effect of the background variation, and iteratively extract and enhance the weak information on the object derived from the motion estimation based on a mixture density model of the background and the motion. Furthermore we developed a generic framework of boundary verification to refine the object boundary by verifying and synthesizing into a more appropriate object contour from the contours obtained independently from other approaches of different reliabilities.

Along with the problem of tracking an object through video frames, the watermarking of the frame images has also been investigated for copyright protection as well as for potentially embedding other useful data into a video sequence. The focus here is mostly on the image watermarking, as its extension or adaptation to a video or even an object region is often considered natural. More precisely we developed a watermarking scheme that embeds the watermark
into the choice of the wavelet filters chosen for the image decomposition, in which selected subbands are processed to mark the watermark existence. These wavelet filters originate from a variety of wavelet families, and they are selected to meet the requirement of sufficiently differentiating one another for a given decomposition path in terms of the resulting subband. Our novel approach here is completely different from the traditional watermarking scheme of fixed set of filters, and is shown to achieve strong robustness that resists a good extent of image distortion by white noises, cropping and rescaling. For watermark attack by the image rotation and rescaling, we developed a scheme that synchronizes the attacked image with the original, based on the significant points in the suspected image, so as to facilitate our proposed detection process. We finally note that, as a possible future pursuit, we may apply our watermarking principle to the copyright protection or other data embedding along a particular object in a video.
REFERENCE


226


[48]. Elgammal, A., Duraiswami, R. and Davis, L. S. “Efficient Kernel Density Estimation Using the Fast Gauss Transform with Applications to Colour Modelling and


Appendix A: Symbols

Object tracking:

$I$ Intensity of an image  
$O$ Object superscript or subscript  
$B,b$ Background superscript or subscript  
$s$ Regional center  
$x,y$ Position  
$X$ Position (vector)  
$d$ Displacement or distance  
$\Delta$ Difference  
$\Omega$ Region (parameters)  
$R,S$ Region (object/background)  
$K,\phi,\varphi$ Kernel function  
$\lambda$ Weight factor  
$M$ Motion  
$d_\theta$ Total segment disparity  
$p$ Probability density or superscript or subscript for the previous frame  
$r$ Radius  
$t$ Time stamp  
$P$ Previous frame  
$D$ Interframe difference data  
$\Gamma$ Contour  
$\rho$ Probability density or confidence

Watermarking:

$f,F,\sigma$ Wavelet filter  
$\tau,\omega$ Threshold  
$c,d$ Subbands  
$P$ Curve  
$h$ Filter coefficients  
$W$ Watermark
Appendix B: Wavelet filters

The coefficients of each wavelet filter are listed sequentially inside a pair of square brackets, with the initial index contained in a pair of trailing parentheses.

Daub4 = [ 0.4829629131445341, 0.836516307378077, 0.2241438680420134, -0.1294095225512603 ] (0);

Daub6 = [ 0.3326705529500825, 0.8068915093110924, 0.4598775021184914, -0.1350110200102546, 0.0854412738820267, 0.0352262918857095 ] (0);

Daub8 = [ 0.2303778133088964, 0.7148465705529154, 0.6308807679398587, -0.0279837694168599, -0.1870348117190931, 0.0308413818355607, 0.0328830116668852, -0.0105974017850690 ] (0);

AntoniniAnalysis =
[ 3.782845550699535e-02, -2.384946501937986e-02, -1.106244044184226e-01, 3.774028556126536e-01, 8.526986790094022e-01, 3.774028556126536e-01, -1.106244044184226e-01, -2.384946501937986e-02, 3.782845550699535e-02 ] (-4);

AntoniniSynthesis =
[ -6.45388826289386e-02, -4.068941760955867e-02, 4.18092273222124e-01, 7.884856164056651e-01, 4.18092273222124e-01, -4.068941760955867e-02, -6.45388826289386e-02 ] (-3);

Villa1810Analysis =
[ 2.885256501123136e-02, 8.244478227504624e-05, -1.57526469076351e-01, 7.67904884691438e-02, 7.589077294537618e-01, 7.589077294537619e-01, 7.67904884691436e-02, -1.57526469076351e-01, 8.244478227504624e-05, 2.885256501123136e-02 ] (-4);

Villa1810Synthesis =
Villa1 Analysis =
[ 3.782845550699535e-02, -2.384946501937986e-02, -1.106244044184226e-01, 3.774028556126536e-01, 8.526986790094022e-01, 3.774028556126537e-01, -2.384946501937986e-02, 3.782845550699535e-02] (-4);

Villa1 Synthesis =
[-6.453888262893856e-02, -4.068941760955867e-02, 4.18092273222124e-01, 7.88485616405651e-01, -4.068941760955867e-02, -6.453888262893856e-02] (-3);

Villa2 Analysis =

Villa2 Synthesis =
[ 1.418215589126359e-02, 6.29231566859828e-03, -1.087373652243805e-01, -6.916271012030040e-02, 4.481085999263908e-01, 8.328475700934288e-01, 4.481085999263908e-01, -6.916271012030040e-02, -1.087373652243805e-01, 6.29231566859828e-03, 245

Brislawn Analysis =
[ 0.037828455506995, 0.023849465019380, 0.110624404418423, 0.377402855612654, 0.852698679009403, 0.377402855612654, -0.110624404418423, -0.023849465019380, 0.037828455506995] (-8);

Brislawn Synthesis =
[ -0.064538882628938, -0.040689417609558, 0.418092273222212, 0.788485616405664, 0.418092273222212, -0.040689417609558, -0.064538882628938, 0.037828455506995] (-4);

Brislawn2 Analysis =
[ 0.026913419, -0.032303352, -0.241109818, 0.054100420, 0.899506092, 0.899506092, 0.054100420, -0.241109818, -0.032303352, 0.026913419] (-4);

Brislawn2 Synthesis =
[ 0.019843545, 0.032303352, 0.241109818, 0.054100420, 0.899506092, 0.899506092, 0.054100420, 0.032303352, 0.019843545] (-4);

Villa2 Synthesis =
[ 1.418215589126359e-02, 6.29231566859828e-03, -1.087373652243805e-01, -6.916271012030040e-02, 4.481085999263908e-01, 8.328475700934288e-01, 4.481085999263908e-01, -6.916271012030040e-02, -1.087373652243805e-01, 6.29231566859828e-03,
Villa3Analysis =
[-1.290777652578771e-01,
  4.769893003875977e-02,
  7.884856164056651e-01,
  7.884856164056651e-01,
  4.769893003875977e-02,
  -1.290777652578771e-01]
(-6);

Villa3Synthesis =
[ 1.891422775349768e-02,
  6.989495243807747e-03,
  -6.72369471890128e-02,
  1.333892255971154e-01,
  6.150507673110278e-01,
  1.333892255971154e-01,
  -6.72369471890128e-02,
  6.989495243807747e-03,
  1.891422775349768e-02]
(-5);

Villa4Analysis =
[-1.767766952966369e-01,
  3.535533905932738e-01,
  1.060660171779821e+00,
  3.535533905932738e-01,
  -1.767766952966369e-01]
(-2);

Villa4Synthesis =
[ 3.535533905932738e-01,
  7.071067811865476e-01,
  3.535533905932738e-01]
(-1);

Villa5Analysis =
[ 7.071067811865476e-01,
  7.071067811865476e-01]
(0);

Villa5Synthesis =
[-8.838834764831845e-02,
  8.838834764831845e-02,
  7.071067811865476e-01,
  7.071067811865476e-01,
  8.838834764831845e-02,
  -8.838834764831845e-02]
(-2);

Villa6Analysis =
[ 3.314563036811943e-02,
  -6.629126073623885e-02,
  -1.76776952966369e-01,
  4.198446513295127e-01,
  9.943689110435828e-01,
  4.198446513295127e-01,
  -1.76776952966369e-01,
  -6.629126073623885e-02,
  3.314563036811943e-02]
(-4);

Villa6Synthesis =
[ 3.535533905932738e-01,
  7.071067811865476e-01,
  3.535533905932738e-01]
(-1);

OdegardAnalysis =
[ 5.2865768532960523e-02,
  -3.3418473279346828e-02,
  -9.3069263703582719e-02,
  3.8697186387262039e-01,
  7.8751377152779212e-01,
  3.8697186387262039e-01,
  -9.3069263703582719e-02,
  -3.3418473279346828e-02,
  5.2865768532960523e-02]
(-4);

OdegardSynthesis =
[ -8.6748316131711606e-02,
  -5.4836926902779436e-02,
  4.4030170672498536e-01,
  8.1678063499210640e-01,
  4.4030170672498536e-01,
  -5.4836926902779436e-02,
  -8.6748316131711606e-02]
(-3);

mna2Analysis =
[ 9.926945048287597E-02,
  3.535533905932737E-01,
  9.056456821522993E-01,
  3.535533905932737E-01,
  -9.926945048287597E-02]
(-2);

mna2Synthesis =
[ 2.542839401103979E-01, 9.056456821522996E-01, 2.542839401103979E-01] (-1);

mna3Analysis =
[ -1.201790019175163E-02, -5.691064014864630E-02, 3.358036623203498E-01, 8.804633184131914E-01, 3.358036623203498E-01, -5.691064014864630E-02, -1.201790019175163E-02] (-3);

mna3Synthesis =
Appendix C: Images and Videos

C.1: Selected frames of test videos

Video 1.

Video 2.
Video 8.

Video 9.
C.2: Test images

Figure 1 Barbara

Figure 2 Goldhill
Figure 3 Peppers

Figure 4 Baboon
Appendix D: Publication List


