Forward-Backward Correlation for Template-Based Tracking

A Thesis

Presented to

the Graduate School of

Clemson University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Electrical Engineering

by

Xiao Wang

May 2006

Advisor: Dr. Stanley T. Birchfield

May 5, 2006

To the Graduate School:

This thesis entitled “Forward-Backward Correlation for Template-Based Tracking” and written by Xiao Wang is presented to the Graduate School of Clemson University. I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

Dr. Stanley T. Birchfield, Advisor

We have reviewed this thesis
and recommend its acceptance:

Dr. Ian D. Walker

Dr. Adam W. Hoover

Accepted for the Graduate School:

Abstract

A significant limitation of traditional template-based trackers is their inability to handle out-of-plane rotation, which can cause total self-occlusion of the target. We present a simple extension to template-based tracking that overcomes this problem. A forward correlation-based search for computing the transformation (displacement and scale) between two image frames is augmented with a backward correlation-based search for grouping the pixels with similar image velocities. In effect, the latter performs motion segmentation on the pixels around the target to automatically update the model as it changes over time, while avoiding drift. A gradient module assists the algorithm when the background contains little texture. Experimental results demonstrate the effectiveness of the technique in both textured and untextured environments, with complete self-occlusion caused by full 360-degree out-of-plane rotation, along with scale changes.

Dedication

To my beloved mom and dad.

Acknowledgments

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

I would like to thank Dr. Walker and Dr. Hoover for gracing my committee with their presence.

I would also like to thank all my friends for their friendship and generous help.

Above all, I would like to thank my parents, my grandparents and my aunt for their constant support, encouragement and love, without which I would never have been able to accomplish this work.

TABLE OF CONTENTS

Page

TITLE PAGE i

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

LIST OF FIGURES vii

1 Introduction 1

1.1 Tracking Framework 2

1.2 Outline of the Thesis 5

2 Template-Based Tracking 6

2.1 Template Selection 6

2.2 Cross Correlation 7

2.3 Motion Model 9

3 Forward-Backward Correlation 11

3.1 Forward Correlation 11

3.2 Drift Problem 13

3.3 Backward Correlation 15

3.3.1 Motivation 15

3.3.2 Overview of Motion Segmentation 15

3.3.3 Backward Correlation 18

4 Untextured Backgrounds 23

4.1 Limitation of Backward Correlation 23

4.2 Motivation 25

4.3 Gradient Module 26

Table of Contents (Continued)

Page

4.4 Combining Modules 28
4.5 Adaptive Scale 30

5 Experimental Results 31

5.1 Experiment on Cluttered Background 31
5.2 Experiment on Untextured Background 35
5.3 Other Experiments 38

6 Conclusion 41

BIBLIOGRAPHY 42

LIST OF FIGURES

Figure   Page

3.1 Drift problem. (a): previous frame. (b): current frame. Templates are marked with solid rectangles. Region A is occluded in the current frame due to rotation. Region C is occluded in the previous frame and comes to the foreground in the current frame. Forward correlation matches region B between two template windows to yield a low SSD error. 14

3.2 Backward correlation search. (a): reference frame $I(\mathbf{x})$.The template drifts away from the object. (b): current image $J(\mathbf{x})$. Forward search is not able to pull the template back to the object. (c): difference image $D(\mathbf{x}) = [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2$. Templates computed by forward search are drawn in rectangles with solid lines. Pixels with similar image velocity generate low SSD errors and are marked with hatchings. Backward correlation chooses the template window (dashed rectangle) that has the lowest SSD error in $D(\mathbf{x})$. 20

4.1 Limitation of backward correlation. (a): difference image $D(\mathbf{x}) = [I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2$ under textured background. Foreground pixels yield much lower SSD errors (marked with hatchings) than background pixels. (b): difference image under untextured background. The region with low SSD errors is marked with hatchings. Foreground and background pixels both yield low SSD errors. Backward correlation has no evidence to prefer the foreground (dashed rectangle) to the background (solid rectangle). 24

4.2 Combination of modules. The base for the tracking framework is forward correlation module which estimates the object's location in the current frame. We need to determine whether the background around the object is textured before we select the next module. 28

5.1 Tracking results of traditional template-based tracker on a sequence with cluttered background. Frames 10, 15, 20, 25, 28 and 30 are shown. The tracker drifts away from the head due to out-of-plane rotation and loses the target completely after several frames. 32

5.2 Tracking results of our algorithm on a sequence with cluttered background. Frame 10, 15, 20, 25, 30, 43, 51, 62, 164, 169, 197 and 239 are shown. The algorithm maintains a tight lock on the target in all the intermediate frames of the out-of-plane rotation. 33

List of Figures (Continued)

Figure Page

5.3 Tracking error of Experiment 1 using our algorithm (solid) vs. the traditional template-based tracker (dashed). (a): tracking error in *x* direction. (b): tracking error in *y* direction. 34

5.4 Tracking results of traditional template-based tracker on a sequence with untextured background. Frames 63, 67, 76, 81, 90 and 102 are shown. The tracker drifts away from the head due to out-of-plane rotation and loses the target completely after several frames. 35

5.5 Tracking results of our algorithm on a sequence with untextured background. Frames 20, 25, 33, 44, 63, 67, 76, 81 and 102 are shown. The tracker is able to handle out-of-plane rotation, along with scale changes. . . 36

5.6 Likelihood of backward correlation module. (a): likelihood at a single scale for frame 25 of Experiment 1. (b): likelihood at a single scale for frame 63 of experiment 2. Backward correlation search starts at point $(0, 0)$. Frame 25 of Experiment 1 has a textured background around the head, and the peak of the likelihood gives the correct displacement. For frame 63 of experiment 2, the background around the head has little texture, and the likelihood is distracted by background pixels to the edge of the search range. 37

5.7 Tracking results on a sequence containing occlusion. (a): the head is partially occluded by the hand. (b): the head is partially occluded by a chair. Frames 34, 35, 40, 41, 55, 140, 143, 148, 152 and 156 are shown. 38

5.8 Tracking results of traditional template-based tracker on a sequence of a vehicle making a turn. 39

5.9 Tracking results of our algorithm on a sequence of a vehicle making a turn. The tracker is kept on the target despite of the significant pose changes of the vehicle. Scale is also well handled. 39

5.10 Tracking results of our algorithm on a sequence which histogram-based algorithm fails. Frames 2, 15. 22, 26, 27 and 28 are shown. 40

Chapter 1

Introduction

Computer vision is the study and application of techniques which enable computer-based systems to *understand* image content and better interact with the real world. Object tracking, as an important computer vision task, is the process of locating single or multiple moving objects in time using a camera. Object Tracking requires the system to automatically extract useful information about the surrounding from the image data in the video sequence. Research in this area has been conducted for various purposes and has led to several important applications:

- Security and surveillance — recognizing people, providing better sense of security using visual information.
- Medical therapy — improving the quality of life for physical therapy patients and disabled people.
- Retail space instrumentation — analyzing shopping behavior of customers, enhancing building and environment design.
- Video abstraction — obtaining automatic annotation of videos, generating object-based summaries.
- Traffic management — analyzing flow to detect accidents.

- Video editing — eliminating cumbersome human-operator interaction, designing futuristic video effects.

Template-based tracking using the sum-of-squared differences (SSD) is a classic technique for maintaining the location of a target throughout an image sequence [7, 10, 13, 14]. The idea of template-based tracking is to track a moving object by defining a region of pixels belonging to that object and, using local optimization methods, to estimate the transformation parameters of that region between the reference image and the new image. The reference image can be fixed as the first frame or chosen to be the previous frame.

Recently, several adaptive approaches have been proposed. Jepson et al. [9] combine modules looking at the short-term and long-term history of the object's appearance, to take advantage of the inherent strength of both recent and older image frames. The relative weighting between the modules is accomplished using expectation-maximization. Ho et al. [8] model the target using a linear subspace created from recent image frames, and the matching in the most recent image is performed using the uniform L^2 reconstruction error. Avidan [2] uses Adaboost to create a strong classifier from several weak classifiers obtained over the recent image frames. Collins et al. [5] adaptively select between different feature spaces on-line in order to maximize discriminative ability and minimize the likelihood of distraction. Yacoob et al. [15] extend polynomial parameterized flow models by adding a structure-compactness constraint that accounts for image motion that deviates from a planar structure.

1.1 Tracking Framework

Birchfield [4] presents a real-time head tracker that makes use of two modules whose failure modes are orthogonal to each other. The tracker works in a way that one module keeps the tracker on the target when the other fails.

The first module is an intensity gradient module that maximizes the gradient around the perimeter of the object (head) being tracked. The head is modeled as a vertical ellipse with a fixed aspect ratio. The coordinates of the center of the ellipse is given by (x, y) and the length of the minor axis is σ. Then the state (location) of the head can be denoted by three parameters $\mathbf{s} = (x, y, \sigma)$. The tracking task is to find the best state $\mathbf{s}$ for each image frame. The elliptical model implies the assumption or constraint that the gradient direction of the boundary is desired to be perpendicular to the perimeter. The likelihood measure is given by the normalized matching score as follows:

$$\phi_g(\mathbf{s}) = \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} \mid \mathbf{n}_\sigma(i) \cdot \mathbf{g}_\mathbf{s}(i) \mid \tag{1.1}$$

where $\mathbf{g}_\mathbf{s}(i)$ is the intensity gradient at perimeter pixel i at location $\mathbf{s}$, $\mathbf{n}_\sigma(i)$ is the unit vector normal to the ellipse at pixel i, and N_σ is the number of pixels on the perimeter of an object with size σ. The local search is performed within a certain range of the predicted position. A constant velocity prediction [3] is employed to compute the predicted position (x^p, y^p) using the positions found in the previous two frames:

$$\begin{aligned} x_t^p &= 2x_{t-1}^* - x_{t-2}^* \\ y_t^p &= 2y_{t-1}^* - y_{t-2}^* \\ \sigma_t^p &= \sigma_{t-1}^*. \end{aligned} \tag{1.2}$$

The second module in [4] is a color histogram module. A module histogram is first constructed off-line. At run time, the histogram intersection [12] is computed between the model histogram M and the image histogram I at each search location. Then the likelihood function is given by:

$$\phi_c(\mathbf{s}) = \frac{\sum_{i=1}^{N} \min(I_\mathbf{s}(i), M(i))}{\sum_{i=1}^{N} I_\mathbf{s}(i)} \tag{1.3}$$

where $I_{\mathbf{s}}(i)$ and $M(i)$ are the numbers of pixels in the ith bin of the histograms, and N is the number of bins.

In order to combine the above two modules, the likelihood is converted to a percentage score by subtracting the minimum and dividing by range. For example, the score for the gradient module is:

$$\bar{\phi}_g(\mathbf{s}) = \frac{\phi_g(\mathbf{s}) - \min_{\mathbf{s}_i \in S} \phi_g(\mathbf{s}_i)}{\max_{\mathbf{s}_i \in S} \phi_g(\mathbf{s}_i) - \min_{\mathbf{s}_i \in S} \phi_g(\mathbf{s}_i)} \tag{1.4}$$

Then the sum of the normalized scores of all modules gives the final likelihood. The the state (location) of the head is thus determined by:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}_i \in S} \{\bar{\phi}_g(\mathbf{s}_i) + \bar{\phi}_M(\mathbf{s}_i)\} \tag{1.5}$$

where $\bar{\phi}_M$ is the score for the color histogram module.

The robust head tracker described above presents us a tracking framework in which multiple modules are employed to overcome the limitation of any single module. If we select the modules in such a way that their assumptions are orthogonal to each other, one module can keep the tracker on the target when the other fails since they have orthogonal failure modules.

The base of the tracking algorithm in this thesis is a template-based tracking method. We explore an adaptive approach to template-based tracking in order to overcome the limitations of the traditional formulations. We augment the standard *forward* correlation based search for the best transformation vector between the template and the current image, with a *backward* correlation search to determine the location of the template in the reference image. In other words, the forward search computes the transformation from the reference image to the current image assuming that the template is correctly positioned in the reference image. The backward search then questions that assumption, finding the best location in the reference image given the computed transformation. The backward search is essentially a motion segmentation module that determines which pixels in the reference image are likely to belong to the target being tracked. The intensity gradient module is performed to assist the algorithm when the background contains little texture.

A significant limitation of traditional template-based trackers is their inability to handle out-of-plane rotation. Such difficult changes in the appearance of the target have led many researchers to explore spatially-invariant features such as color histograms as an alternative to templates. However, histograms lack the spatial information and the specificity that is available with more detailed models such as templates. This thesis focuses on the improvement of template-based approaches. We consider histogram-based approaches as complimentary methods that can be incorporated into our tracking framework in the future research.

1.2 Outline of the Thesis

In this thesis, we aim to improve the performance of traditional template-based tracking. Chapter 2 reviews the traditional template-based tracking algorithm. We examine the issues in choosing template and motion model. We also discuss the correlation coefficient and its advantages in cross correlation. In Chapter 3, we first implement the standard template-based tracking algorithm which we refer to as forward correlation search. The drift problem is then discussed and the cause of such problem is studied. A backward correlation module is proposed to overcome this limitation of the standard forward correlation. Chapter 4 further analyzes the limitation of the backward correlation and introduce the intensity gradient module to assist the tracking algorithm under untextured environment. Experimental results are shown in Chapter 5 and conclusions are drawn in Chapter 6.

Chapter 2

Template-Based Tracking

The goal of template-based tracking is to maintain a model of the target in terms of a 2D template of image intensities and compute the target location in a new image frame by comparing the new data with that of the template. The data are usually compared using a low-order parametric motion model such as translation or affine, and the optimal location is computed using either discrete correlation search or non-linear function optimization. Template-based matching algorithm is usually simple, effective and computationally efficient.

2.1 Template Selection

A critical question in template-based tracking is how to select the template. One approach is to use the appearance of the target in the first image frame, with the template remaining constant throughout the sequence. The advantage of this approach is that the tracker always uses data which is known to be trustworthy. However, the drawback is that the algorithm does not adapt to changes in the appearance of the target over time. The target is likely to be lost when it rotates out of plane or undergoes non-rigid transformations. An alternative approach is to use the appearance of the target in the previous image frame, so that the tracker always adapts to changes in appearance. The disadvantage of this approach is that the tracker tends to drift away from the original target over time, since there is no guarantee that the newly computed location of the target is without error.

This *reference frame* dilemma is not limited to template-based tracking: Histogram trackers are also faced with the choice of which image to use as a reference, with many of them selecting the first frame [4, 6].

2.2 Cross Correlation

Cross correlation is an algorithm for the location of corresponding image patches based on the similarity of gray levels. A reference point is given in the reference image, and its coordinates are searched for in the current image. For that purpose, the reference image is moved in the search image, and the position of maximum similarity of gray levels is searched for. At each position of the reference image in the search image, a similarity value is calculated.

The use of cross-correlation for template matching is motivated by the SSD distance measure (i.e., squared Euclidean distance),

$$d_{t,f}^2(\Delta x, \Delta y) = \sum_{x,y}[t(x,y) - f(x+\Delta x, y+\Delta y)]^2 \tag{2.1}$$

where f is the search image and t is the reference image. Δx and Δy are the displacement between the reference image and the search image in the x and y directions, respectively. The sum is taken over all the pixels of the region of interest.

From equation (2.1), We can expand $d_{t,f}^2$ as:

$$d_{t,f}^2(\Delta x, \Delta y) = \sum_{x,y}[t^2(x,y) - 2t(x,y)f(x+\Delta x, y+\Delta y) + f^2(x+\Delta x, y+\Delta y)] \tag{2.2}$$

In equation (2.2), the term $\sum t^2(x,y)$ is constant. If the term $\sum f^2(x+\Delta x, y+\Delta y)$ is approximately constant, then the remaining term

$$s(\Delta x, \Delta y) = \sum_{x,y} t(x,y) f(x+\Delta x, y+\Delta y) \tag{2.3}$$

is a measure of the similarity between the template and the search image.

However, there are several disadvantages if we use equation (2.3) for template matching.

- Equation (2.3) is drawn under the assumption that the image energy is approximately constant. If the energy $\sum f^2(x,y)$ varies with position, matching using equation (2.3) can fail. For example, the correlation between the template and the exactly matching region in the image may be less than the correlation between the template and a bright spot.

- The range of $s(\Delta x, \Delta y)$ is dependent on the size of the template window. This will cause inconvenience when used for tracking object with scale change.

- Equation (2.3) is not invariant to changes in image amplitude such as changes caused by changing lighting conditions across the video sequence.

To overcome these difficulties and disadvantages, the *correlation coefficient* is introduced by normalizing the image and template to unit length:

$$c(\Delta x, \Delta y) = \frac{\sum_{x,y}[t(x,y)-\bar{t}][f(x+\Delta x, y+\Delta y)-\bar{f}]}{\sqrt{\sum_{x,y}[t(x,y)-\bar{t}]^2 \sum_{x,y}[f(x+\Delta x, y+\Delta y)-\bar{f}]^2}} \tag{2.4}$$

where $\bar{t}$ is the mean gray level of the template and $\bar{f}$ is the mean of the image region f compared with the template.

2.3 Motion Model

There are a number of ways of defining the relationship between one reference system and another. The choice of the most appropriate motion transformation model is influenced by such factors as:

- Whether the model is to be applied to a small area, or over a large region.
- Whether the target has significant distortion.
- The accuracy required.
- Whether the transformation parameters are available, or must be determined.

An affine transformation is an important class of linear 2-D geometric transformations which maps variables into new variables by applying a linear combination of translation, rotation, scaling, and/or shearing operations. An affine transformation transforms straight lines to straight lines and parallel lines remain parallel. Generally the size, shape, position, and orientation of lines will change. The scale factor depends on the orientation of the line but not on its position within the coordinate system. Hence the lengths of all lines in a certain direction are multiplied by the same scalar. The general affine transformation can be commonly written in homogeneous coordinates as shown below:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = A \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + B \qquad (2.5)$$

where (x_1, y_1) is the position of an image pixel in the input image and (x_2, y_2) is the position of corresponding pixel in the output image.

Pure translation can be carried out by defining:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad (2.6)$$

Considering positive angles as clockwise rotations, pure rotation can be defined as:

$$A = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.7}$$

Pure scaling is:

$$A = \begin{bmatrix} \alpha_{11} & 0 \\ 0 & \alpha_{22} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.8}$$

Since an affine transformation is equivalent to the composed effects of translation, rotation, scaling and/or shearing, we can combine the above transformations to achieve an resultant transformation. Note that the order in which the transformations occur is significant since a translation followed by a rotation is not necessarily equivalent to the converse.

A transformation in which the scale factor is the same in all directions is called a *similarity transformation*. A similarity transformation preserves shape, so angles do not change, but the lengths of lines and the position of points may change.

Assuming that the object is rigid and compact, a similarity transformation is a reasonable approximation of the motion model. We can further assume that the object of interest undergoes motion between two consecutive image frames that is well-approximated by translation and scaling. In practice, this approximation works well even for objects undergoing more complex motion, such as rotation, as long as the frame rate of the camera is high compared with the object's motion.

Chapter 3

Forward-Backward Correlation

In this chapter, we first implement the standard template-based tracker which we refer to as forward correlation module. The drift problem is then discussed and the cause of such problem is studied. A backward correlation module is therefore proposed to overcome this limitation of the standard forward correlation.

3.1 Forward Correlation

In forward correlation search, the traditional template matching technique is used to find the approximate location of the target in the current image. We choose the appearance of the target in the previous image frame to be the template, so that the tracker can adapt to changes in appearance.

Let $\Omega(t)$ denote the estimated target region at any time t. Every point $\mathbf{x} = (x, y)$ in the target region $\Omega(t)$ can be obtained from a corresponding point $\mathbf{x}' = (x', y')$ in the template Ω_0 via a coordinate transformation $\varphi : \Omega_0 \mapsto \Omega(t)$ as follows:

$$\mathbf{x} = \varphi(\mathbf{x}';\ \mathbf{d}(t)) \tag{3.1}$$

where $\mathbf{d}(t)$ denotes the transformation that specifies the location of the object in the current image frame. Assuming the object is rigid and compact, a similarity transform $\mathbf{d} = (d_x, d_y, \alpha)$ yields a reasonable approximation, where (d_x, d_y) is the displacement in the x and y directions, respectively, and α is the scaling. In this case the transformation can be rewritten as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \alpha \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}. \tag{3.2}$$

Let $I(\mathbf{x}, t)$ denote the intensity of pixel $\mathbf{x}$ at time t. Let $\mathcal{W}(t-1)$ denote the set of pixels (i.e., the template) corresponding to the object in the previous image $I(\mathbf{x}, t-1)$, at time $t-1$. Then the location of the object in the current frame $I(\mathbf{x}, t)$, can be found by locally searching for the transformation vector $\mathbf{d}$ that minimizes a sum-of-squared difference (SSD) error:

$$\mathbf{d}^* = \arg\min_{\mathbf{d}\in\mathcal{D}} \sum_{\mathbf{x}\in\mathcal{W}(t-1)} [I(\mathbf{x}, t-1) - I(\varphi(\mathbf{x};\ \mathbf{d}), t)]^2 \tag{3.3}$$

where $\mathcal{D}$ is the set of transformation vectors being considered. This equation is the standard formulation for basic template-based tracking that we refer to as *forward* correlation.

To overcome the difficulties caused by such as changes of illumination and size of the template, we can put equation (3.3) into correlation coefficient framework described in equation (2.4). The correlation coefficient is:

$$c(\mathbf{d}) = \frac{\sum_{\mathbf{x}\in\mathcal{W}(t-1)} [I(\mathbf{x}, t-1) - \bar{I}_{t-1}][I(\varphi(\mathbf{x};\ \mathbf{d}), t) - \bar{I}_t]}{\sqrt{\sum_{\mathbf{x}\in\mathcal{W}(t-1)} [I(\mathbf{x}, t-1) - \bar{I}_{t-1}]^2 \sum_{\mathbf{x}\in\mathcal{W}(t-1)} [I(\varphi(\mathbf{x};\ \mathbf{d}), t) - \bar{I}_t]^2}} \tag{3.4}$$

where $\bar{I}_{t-1}$ is the mean gray value of the template and $\bar{I}_t$ is the mean part of the current image compared with the template. We can see from equation (3.4) that the correlation coefficient $c(\mathbf{d})$ is a function of the transformation vector $\mathbf{d}$. By moving the template in the search region in the current image, equation (3.4) actually generates a likelihood map of the location of the target. The likelihood map gives the probabilities of the target to be at a particular position. The size of the likelihood map is determined by the size of the local search window. The location of the target in the current image frame is therefore the location with the largest likelihood in the likelihood map:

$$\mathbf{d}^* = \arg\max_{\mathbf{d}\in\mathcal{D}} c(\mathbf{d}) \tag{3.5}$$

In forward correlation, the above exhaustive local search is performed by varying the position of the target as long as the scale. In implementation, for the sake of computational efficiency, we vary the scale by ± 10 percent for every position within the search window. Once the best transformation vector $\mathbf{d}^*$ is found, the template is shifted by it to yield the object's location at time t: $\mathcal{W}(t) = \varphi(\mathcal{W}(t-1);\ \mathbf{d}(t))$. Then $\mathcal{W}(t)$ is the template to be used in frame $t+1$. The template is thus updated each frame during the tracking procedure.

3.2 Drift Problem

In forward correlation search described in Section (3.1), the traditional template matching technique is used for tracking. The template for the first image frame in the video sequence is manually selected, then the template is updated each frame by the forward correlation algorithm. For each image frame, a likelihood map is generated from the correlation between the template and the local search region in the current image. For each location within the search region, the likelihood map gives a probability of the object to be at such location. The location of the object in the current frame is then determined by the location with the highest value in the likelihood map. Since there is no guarantee that the newly computed location of the object is without error, it is highly likely that the template may contain false information about the true appearance of the target, i.e., the template may include some of the background pixels. The forward correlation algorithm does not check the reliability of the template, so the error is likely to accumulate over time.

Experiments show that forward correlation search is generally efficient when dealing with pure translation. The tracker can also handle scale change along with the motion.

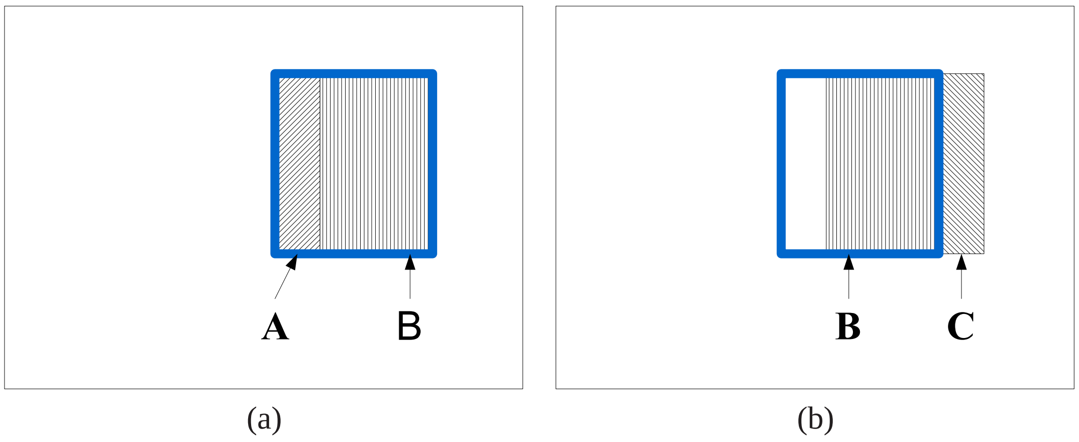


Figure 3.1: Drift problem. (a): previous frame. (b): current frame. Templates are marked with solid rectangles. Region A is occluded in the current frame due to rotation. Region C is occluded in the previous frame and comes to the foreground in the current frame. Forward correlation matches region B between two template windows to yield a low SSD error.

However, the tracker will drift away from the object when the object is rotating out of plane because of self-occlusion. Out-of-plane rotation is a frequent behavior in object tracking, especially in head tracking. A person turning around is a good example.

The cause of such drift problem is also shown in Figure (3.1). The object performs out-of-plane rotation between these two consecutive frames. Region A of the object is occluded in the current image due to the rotation; region C, which is occluded in the previous frame, comes to the foreground in the current image. We can see that the result of out-of-plane rotation is the change of the appearance of the object. Suppose that the template is correctly overlaid on the object in the previous frame. The forward search maximizes the correlation between the template window in the two images. So the lowest SSD error is obtained when region B is placed in the same location in the new template window compared to the old template.

To summarize, the forward correlation approximates the rotation with translation parallel to the image plane, and it creates small errors in each image. Such errors may accumulate to cause the tracker to lose the target completely after several frames.

3.3 Backward Correlation

3.3.1 Motivation

The previous section addresses the limitation of the forward correlation algorithm. The drift problem is a major reason that the traditional template-based tracking method is not reliable when dealing with long video sequences consisting of complicated motion such as out-of-plane rotation. When the tracker drifts away from the target, the template contains part of the target and part of the background. The forward correlation algorithm performs local search on the current image frame using the template obtained from the previous image frame. Such local search *only* finds the location that maximizes the correlation between the template window and the target window. The resultant location computed is likely also drifted away from the true location since the template used itself is not fully correct. The forward correlation algorithm always assumes that the template is correctly positioned in the reference frame and the algorithm itself does not check the reliability of the template. So here we need a mechanism to question this assumption and make proper adjustment to the template whenever the tracker drifts.

3.3.2 Overview of Motion Segmentation

To overcome the drift problem, we need to check the correctness of the template for each image frame. In other words, we need to determine which pixels in the reference image are likely to belong to the target being tracked. We can consider this as a motion segmentation problem. The main goal of image segmentation is to divide an image into parts that have a strong correlation with objects or areas of the real world depicted in the image. Image segmentation can be divided into three groups: thresholding, edge-based segmentation and region-based segmentation. Segmentation of video sequences is a more complicated task since videos are image sequences of frames, and time has to be taken into consideration. In other words, motion segmentation needs to deal with change in the scene due to motion over time. Therefore, the criterion of common motion is of great significance in motion segmentation.

The main goal of motion segmentation is to group homogeneous image regions based on similar motion. Motion segmentation is a very important topic in motion analysis and is of fundamental importance in detection, tracking, robot navigation, etc. As in many other areas of computer vision, there is no foolproof technique or general algorithm in motion segmentation. Motion segmentation is difficult for several reasons:

- Each image in the video sequence is a projection of the 3-D motion mapped onto a 2-D image plane.This makes the motion parameter solving problem under-constrained and thus additional constraints need to be applied to the motion of the image regions of interest.
- The occlusion and disocclusion make it difficult to accurately associate moving objects. This may occur if an object moves to the background and gets occluded by the foreground object for a while and then reappears to the foreground again.
- Image noise may add to the ambiguity of motion segmentation. A good example for this is that image noise can result in changes of the pixel intensity in the subsequent frame even if there is no motion between two frames.

The different techniques of motion segmentation can be divided into two categories: videos with static camera and videos with moving camera. A static camera allows us to partition the images into foreground and background. Differential motion analysis methods can be applied under such situation. Suppose that an image of a static scene is available and only stationary objects are present in the scene. If we use this image for reference, the difference image suppresses all motionless areas and any motion in the scene can be detected as areas corresponding to the actual positions of the moving objects in the scene. Then motion analysis is based on a sequence of difference images. It is obvious that this technique will fail for videos taken by moving cameras, because the complete image may be changing. Since a static camera can be regarded as a special case of a moving camera, the techniques for videos taken by moving camera are of more general and practical meaning, especially for the task of object tracking.

Generally speaking, motion segmentation can be considered as a two-step procedure:

1. Determine the motion vectors associated with each pixel or feature point.

2. Group pixels or feature points that perform common motion.

Using sparse feature points has the advantage of reducing computing time and computational complexity, compared to using dense motion field to find out motion vectors associated with each pixel. Therefore, detecting and tracking those feature points is the basis for motion segmentation task.

Shi and Tomasi [10] described an algorithm for detecting and tracking feature points, which has been implemented in the Kanade-Lucas-Tomasi (KLT) feature tracker [1]. The dissimilarity of feature windows in two images is given by:

$$\epsilon = \int\int_W [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2 d\mathbf{x} \tag{3.6}$$

The idea of feature tracking here is to minimize the dissimilarity shown in equation (3.6). [10] proposed the criterion to select good features. Consider an image sequence $I(x, t)$ where, $\mathbf{x} = [\mathbf{x}, \mathbf{y}]^{\mathbf{T}}$ are the coordinate of an image pixel. During tracking, it is assumed that intensities of the points in images remain unchanged:

$$I(\mathbf{x}, t) = I(\varphi(\mathbf{x};\ \mathbf{d}), t + \Delta t) \tag{3.7}$$

where φ refers to the motion field. If the image pixels are assumed to be translating then the motion field is specified by $\varphi = \mathbf{x} + \mathbf{d}$ where $\mathbf{d}$ is the linear displacement vector. The aim here is to compute $\mathbf{d}$ such that it minimizes the Sum of Squared Distances (SSD) formulated in equation (3.6).The resultant system is

$$Z\mathbf{d} = \mathbf{e} \tag{3.8}$$

where $Z = \int\int_W \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^T d\mathbf{x}$, $\mathbf{e} = \int\int_W [I(\mathbf{x}) - J(\mathbf{x})]\mathbf{g}(\mathbf{x}) d\mathbf{x}$ and $g(\mathbf{x}) = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]^T$.

Equation (3.8) is solved iteratively for each pair of consecutive images. After the features are detected and tracked, the features with similar motion are grouped together.

3.3.3 Backward Correlation

KLT feature tracker gives us a framework to detect and track features which can be used in motion segmentation. However, we need to notice that KLT feature tracker has its limitation if applied in our object tracking scenario. KLT feature tracker assumes *mutual correspondence*[11], i.e. rigid objects exhibit stable pattern points and each point of an object corresponds to exactly one point in the next image in sequence and vice versa. Hence it assumes that there is no occlusion of the feature window in the next image frame. Any occlusion will result in the loss of that feature. The drift problem we are trying to handle is largely caused by out-of-plane rotation, and such rotation may cause total self-occlusion of the target.

Recall equation (3.6):

$$\epsilon = \int\int_W [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2 d\mathbf{x} \tag{3.9}$$

For a *good* feature [10], the error ϵ minimizes if we find the proper displacement vector d for that feature. In Forward Correlation search, we always assume that the template is correctly positioned in the reference image. Recall equation (3.3):

$$\mathbf{d}^* = \arg\min_{\mathbf{d}\in\mathcal{D}} \sum_{\mathbf{x}\in\mathcal{W}(t-1)} [I(\mathbf{x}, t-1) - I(\varphi(\mathbf{x};\ \mathbf{d}), t)]^2 \tag{3.10}$$

All the pixels in the template $\mathcal{W}(t-1)$ are assumed to correspond to the object being tracked in frame $t-1$. When the tracker drifts away from the object of interest, the template includes part of the background pixels and part of the object being tracked. Now consider the dissimilarity ϵ computed under the template, where in equation (3.6) W is the template, $I(\mathbf{x})$ is the reference image, $J(\mathbf{x})$ is the current image frame, and $\mathbf{d}$ is the displacement vector computed by forward correlation search. The background is moving at a different velocity than that of the object being tracked, the part of background region included in the template window $\mathcal{W}(t-1)$ is not likely to be included in the new template window $\mathcal{W}(t)$ in the current frame.

Here we can decompose the template window into two partitions: The part overlaid on the foreground $\mathcal{W}_f$ and the the part overlaid on the background $\mathcal{W}_b$. So we have:

$$\mathcal{W} = \mathcal{W}_f + \mathcal{W}_b. \tag{3.11}$$

Equation (3.6) can be rewritten as:

$$\epsilon = \int\int_{\mathcal{W}_f} [I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2 d\mathbf{x} + \int\int_{\mathcal{W}_b} [I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2 d\mathbf{x}. \tag{3.12}$$

When the template is correctly overlaid on the object being tracked, $\mathcal{W} = \mathcal{W}_f$ and $\mathcal{W}_b = \emptyset$, thus the second term at the right side of equation (3.12) is zero. When the template is drifted away from the correct position and overlaid only on part of the object being tracked, $\mathcal{W}_b \neq \emptyset$. The window $\mathcal{W}_f$ in the reference image $I(\mathbf{x})$ and the current image $J(\mathbf{x})$ correspond to the foreground of the object being tracked. Since the background and foreground have different image velocities, the window $\mathcal{W}_b$ in the reference image $I(\mathbf{x})$ and the current image $J(\mathbf{x})$ include actually *different* parts of the background. In other words, the displacement vector d here represents the motion of the foreground object, so the dissimilarity of the foreground $\int\int_{\mathcal{W}_f} [I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2 d\mathbf{x}$ generates a low SSD error; d is not the correct displacement for the background pixels included in $\mathcal{W}_b$ in the reference

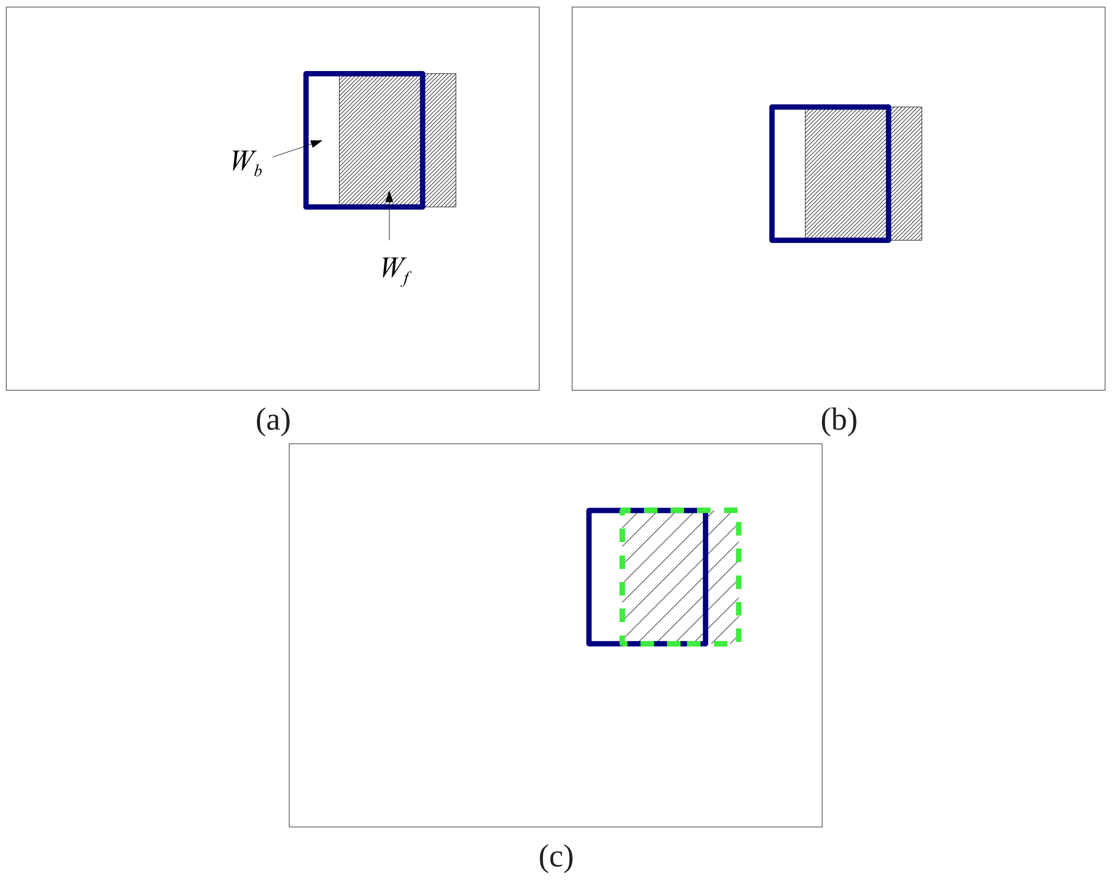


Figure 3.2: Backward correlation search. (a): reference frame $I(\mathbf{x})$.The template drifts away from the object. (b): current image $J(\mathbf{x})$. Forward search is not able to pull the template back to the object. (c): difference image $D(\mathbf{x}) = [I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2$. Templates computed by forward search are drawn in rectangles with solid lines. Pixels with similar image velocity generate low SSD errors and are marked with hatchings. Backward correlation chooses the template window (dashed rectangle) that has the lowest SSD error in $D(\mathbf{x})$.

image, so the dissimilarity of the background $\int\int_{\mathcal{W}_b}[I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d})]^2 d\mathbf{x}$ is likely to generate a high SSD error, and this will result in a high dissimilarity error ϵ.

In forward correlation and feature tracking algorithm, d is considered as the variable to be solved under the same template window or feature window. Since the displacement d is computed by the forward correlation search, we can treat d as a constant and consider $\mathcal{W}$ the variable. From the discussion above, we can see that the dissimilarity under $\mathcal{W}_f$ is much lower compared to the dissimilarity under $\mathcal{W}_b$ and thus the closer the template window is positioned around the correct location of the object being tracked, the lower the final error ϵ is. The pixels of the object being tracked have similar motion and thus similar image velocities between two consecutive image frames, and the displacement d is a good approximation of such motion. When we minimize the dissimilarity ϵ by fixing d and changing $\mathcal{W}$, we also minimize the second term at the right side of equation (3.12).Therefore, the template window $\mathcal{W}$ corresponding to the lowest ϵ is closest to the ideal correct template overlaid on the object being tracked:

$$\mathcal{W} = \arg\min_{\mathcal{W}_i \in \mathcal{V}} \int\int_{\mathcal{W}_i} [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2 d\mathbf{x} \tag{3.13}$$

where $\mathcal{V}$ is the set of the template windows under consideration. This procedure is shown in Figure (3.2).

Now introduce this procedure into our tracking scenario and use the notations defined in Section (3.1):

$$\mathcal{W}(t-1) = \arg\min_{\mathcal{W} \in \mathcal{V}} \sum_{\mathbf{x} \in \mathcal{W}} [I(\mathbf{x}, t-1) - I(\varphi(\mathbf{x};\ \mathbf{d}^*), t)]^2 \tag{3.14}$$

where $\mathbf{d}^*$ is the transformation vector computed by equation (3.3) in forward correlation. The template candidates in the set $\mathcal{V}$ are located around $\mathcal{W}(t-1)$ and, for each location, we vary the scale by ± 10 percent. This equation is the standard formulation for what we refer as *backward* correlation. Similar to forward correlation, we put equation (3.3) into correlation coefficient framework. The correlation coefficient is:

$$c(\mathcal{W}) = \frac{\sum_{\mathbf{x} \in \mathcal{W}} [I(\mathbf{x}, t-1) - \bar{I}_{t-1}][I(\varphi(\mathbf{x};\ \mathbf{d}^*), t) - \bar{I}_t]}{\sqrt{\sum_{\mathbf{x} \in \mathcal{W}} [I(\mathbf{x}, t-1) - \bar{I}_{t-1}]^2 \sum_{\mathbf{x} \in \mathcal{W}} [I(\varphi(\mathbf{x};\ \mathbf{d}^*), t) - \bar{I}_t]^2}} \tag{3.15}$$

where $\bar{I}_{t-1}$ and $\bar{I}_t$ are the mean gray values of the pixels overlaid by the template window in frame $t-1$ and frame t, respectively. The correlation coefficient $c(\mathcal{W})$ is a function of the template $\mathcal{W}$. By moving the template window in the reference image, equation (3.15) actually generates a likelihood map of the template candidates. The likelihood map gives the probabilities of a template window to be overlaid correctly on the target being tracked. The template correctly overlaid on the target in the reference image is therefor the one with the largest likelihood in the likelihood map:

$$\mathcal{W}(t-1) = \arg\max_{\mathcal{W}\in\mathcal{V}} c(\mathcal{W}) \tag{3.16}$$

Once $\mathcal{W}(t-1)$ is computed, the template in the current image $\mathcal{W}(t)$ is determined since we already know the motion vector $\mathbf{d}^*$:

$$\mathcal{W}(t) = \varphi(\mathcal{W}(t-1);\ \mathbf{d}(t)) \tag{3.17}$$

To summarize, backward correlation examines the assumption of forward correlation and finds the best template that maximizes the correlation within the template window in the reference image and the current image. Backward correlation is essentially a motion segmentation module that determines which pixels in the reference image are likely to belong to the target being tracked and then group them together.

Chapter 4

Untextured Backgrounds

In this chapter, we further analyze the tracking mechanism of backward correlation and its limitation under untextured backgrounds. A module based on intensity gradients is proposed to assist the forward correlation search when the background contains little texture. Then we discuss the procedure of combining the three modules (forward correlation module, backward correlation module and gradient module). Scale handling avoiding over-sensitive adaptation is also discussed.

4.1 Limitation of Backward Correlation

The goal of backward correlation is to group the foreground pixels which are moving at the similar image velocities. At the foreground velocity, the dissimilarity of the foreground will always yield a low SSD error. Normally, as discussed in Section (3.3), when we perform backward correlation, the background will yield a high SSD error because it is moving at a different velocity than that of the object being tracked. Recall equation (3.12):

$$\epsilon = \iint_{\mathcal{W}_f} [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2 d\mathbf{x} + \iint_{\mathcal{W}_b} [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2 d\mathbf{x} \tag{4.1}$$

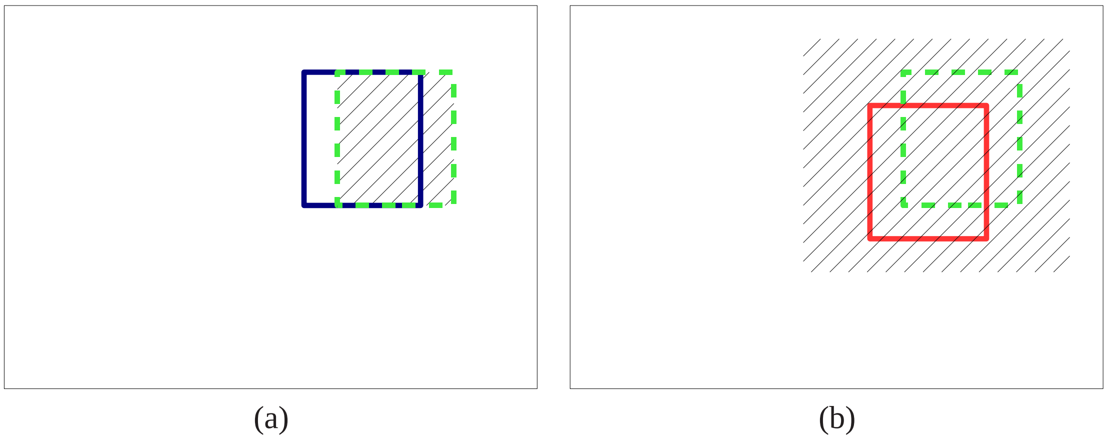


Figure 4.1: Limitation of backward correlation. (a): difference image $D(\mathbf{x}) = [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2$ under textured background. Foreground pixels yield much lower SSD errors (marked with hatchings) than background pixels. (b): difference image under untextured background. The region with low SSD errors is marked with hatchings. Foreground and background pixels both yield low SSD errors. Backward correlation has no evidence to prefer the foreground (dashed rectangle) to the background (solid rectangle).

The idea is to choose the template which minimizes the dissimilarity ϵ. In other words, backward correlation repels the template from the background, thus keeping it on the foreground.

Now let us examine the assumption of backward correlation algorithm. In Section (3.3.3), we state that the correlation between the background pixels using foreground velocity will generate high SSD error. This is under the assumption that different part of the background does not correspond to each other well. However, if the background has little texture, $\int\int_{\mathcal{W}_b}[I(\mathbf{x}) - J(\mathbf{x} + \mathbf{d})]^2 d\mathbf{x}$ will always yield a low SSD error no matter if $\mathbf{d}$ is the correct displacement vector for the template in the reference image. When this happens, the backward correlation algorithm will have no reason to prefer the foreground to the background (shown in Figure (4.1)). In such case, the tracker will be distracted by the background and drift from the target being tracked.

4.2 Motivation

Birchfield [4] proposes that robust, reliable tracking algorithm in a complex environment will require the integration of different modules, making use of different criteria. In order to overcome the limitation of the backward correlation algorithm, we need to employ such a module that its assumptions are, as much as possible, orthogonal to those of the backward correlation algorithm. Thus when the backward correlation module fails, the other module can come to its aid.

The object of interest can be decomposed into two disjoint sets: the boundary and the interior. These two sets are complementary in the mathematical sense, so the failure modes of a tracking module focusing on the object's boundary should be orthogonal to those of a module focusing on the object's interior. Since the forward-backward correlation module matches the intensity values of the object's interior, we are motivated to seek a module focusing on the boundary of the target being tracked.

Edges are often used in image analysis for finding region boundaries. Provided that the region has homogeneous brightness, its boundary is at the pixels where the image function varies and consists of pixels of high edge magnitudes, in the ideal case without noise. An edge is a property attached to an individual pixel and is calculated from the image function behavior in a neighborhood of that pixels. It is a vector with two components, magnitude and direction. The edge magnitude is the magnitude of the gradient, and the edge direction ϕ is rotated with respect to the gradient direction ψ by $-90°$. The gradient direction gives the direction of maximum growth of the function, e.g., from black ($I(\mathbf{x}) = 0$) to white ($I(\mathbf{x}) = 255$).

The gradient magnitude $\mathbf{g}(x, y)$ are continuous image functions calculated as:

$$\mid \mathbf{g}(x, y) \mid = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{4.2}$$

and gradient direction ψ is:

$$\psi = \arg\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{4.3}$$

where $\arg(x,y)$ is the angle from the x axis to the point (x,y).

To summarize, the limitation of the forward-backward correlation algorithm motivates us to utilize an edge-based module focusing on the boundary of the object being tracked other than the interior.

4.3 Gradient Module

To goal of the gradient module, similar to that of the backward correlation module, is to make proper adjustment to the result of the forward correlation module. Provided that the template is drifted from the target being tracked, the boundary of the template also deviates from the boundary of the object's true boundary. So a proper edge-based segmentation of the object of interest is desired here.

Edge-based segmentation represents a large group of methods based on information about edges in the image and it is one of the earliest segmentation approaches and still remains very important [11]. Edge-based segmentation rely on edges found in an image by edge detecting operators. Edges mark image locations of discontinuities in gray level, color, texture, etc. The aim of edge-based segmentation to achieve at least a partial segmentation, which is to group local edges into an image where only edge chains with a correspondence to existing objects are present.

An important factor in edge-based segmentation is the prior information that we can use to incorporate into the method. Prior information affects the segmentation algorithm. Generally, the more prior information that is available to the segmentation process, the better the segmentation can be obtained. Prior information can also be included in the confidence evaluation of the resulting segmentation as well. If much prior information about the desired result is known, the boundary shape and relations with the other image structure are specified very strictly and the segmentation must satisfy all those specifications.

For our object tracking scenario, the shape of the object of intest is assumed to be already known. For example, we can use an ellipse to model the shape of the head in head tracking. This prior information is important to the segmentation process because the segmentation result for the head in the image is thus specified to be an ellipse. Therefore, we can evaluate the goodness of match around the boundary of an object by computing the normalized sum of the gradient magnitude around its perimeter. We have the measurement function as follows:

$$\phi_g(\mathbf{s}) = \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} \mid \mathbf{g_s}(i) \mid \tag{4.4}$$

where $\mathbf{g_s}(i)$ is the intensity gradient at perimeter pixel i at location $\mathbf{s}$, and N_σ is the number of pixels on the perimeter of an object with size σ.

Equation (4.4) only desires large gradient magnitudes around the perimeter. A more accurate measure also desires the gradient direction to be perpendicular to the perimeter:

$$\phi_g(\mathbf{s}) = \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} \mid \mathbf{n}_\sigma(i) \cdot \mathbf{g_s}(i) \mid \tag{4.5}$$

where $\mathbf{n}_\sigma$ is the unit vector normal at pixel i, and $(\cdot)$ denotes the dot product. We use the gradient dot product due to its improved performance.

The gradient module alone performs well in untextured environments tracking a person walking around an untextured room and it fails under cluttered environment[4]. Since the backward correlation module and gradient module have orthogonal failure modes, they can complement each other under either textured or untextured environment.

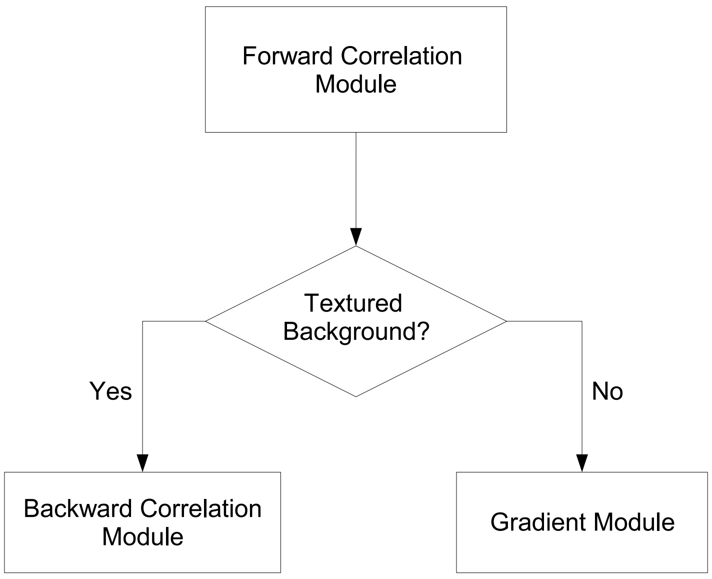


Figure 4.2: Combination of modules. The base for the tracking framework is forward correlation module which estimates the object's location in the current frame. We need to determine whether the background around the object is textured before we select the next module.

4.4 Combining Modules

After forward correlation search is performed, the object's location has been estimated in the current frame. To decide which of the two modules (backward correlation module or gradient module) should be employed, we need to determine whether the background around the head is textured. Our solution is to use the sum of the gradient magnitude of the neighborhood region of the location of the head obtained via forward search. The sum is then thresholded to determine whether the background is textured. Backward correlation module is chosen when the background is cluttered; gradient module is employed when the background contains little texture. This process is shown in Figure (4.2).

The combination of the forward correlation module and backward correlation module is straightforward. Backward correlation is performed after forward correlation using the translation vector (displacement and scale) computed in the forward search. This procedure finds not only the best displacement between the images, but also the best location of the template in the previous image, which has the effect of improving the estimate of the target in the current image.

Since the forward correlation and gradient modules are both used to compute the displacement, combining them requires the normalization of their matching scores. Recall equation (3.4), the likelihood function of forward search is given by the correlation coefficient:

$$\phi_c(\mathbf{d}) = c(\mathbf{d}) \tag{4.6}$$

Define $f : \mathcal{S} \mapsto \mathcal{D}$ which computes the transformation vector $\mathbf{d}$ given the search state $\mathbf{s}$ and the state $\mathbf{s}_{t-1}$ in the previous frame. If we let $\mathbf{s} = (x, y, \sigma)$ and $\mathbf{d} = (d_x, d_y, \alpha)$ (Section (3.1)), we have the following:

$$\begin{aligned} d_x &= x - x_{t-1} \\ d_y &= y - y_{t-1} \\ \alpha &= \sigma / \sigma_{t-1}. \end{aligned} \tag{4.7}$$

Rewrite equation (4.6) so that the forward correlation score is a function of the search state $\mathbf{s}$:

$$\phi_c(\mathbf{s}) = c(f(\mathbf{s})) \tag{4.8}$$

As described in [4], to facilitate adding forward correlation score with the gradient score, the former is converted to a percentage by subtracting the minimum and dividing by the range:

$$\bar{\phi}_c(\mathbf{s}) = \frac{\phi_c(\mathbf{s}) - \min_{\mathbf{s}_i \in S} \phi_c(\mathbf{s}_i)}{\max_{\mathbf{s}_i \in S} \phi_c(\mathbf{s}_i) - \min_{\mathbf{s}_i \in S} \phi_c(\mathbf{s}_i)} \tag{4.9}$$

The gradient score is also converted to a percentage in a similar way:

$$\bar{\phi}_g(\mathbf{s}) = \frac{\phi_g(\mathbf{s}) - \min_{\mathbf{s}_i \in S} \phi_g(\mathbf{s}_i)}{\max_{\mathbf{s}_i \in S} \phi_g(\mathbf{s}_i) - \min_{\mathbf{s}_i \in S} \phi_g(\mathbf{s}_i)} \tag{4.10}$$

Then the final state is decided by combining the two normalized scores:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}_i \in S} \{\bar{\phi}_c(\mathbf{s}_i) + \bar{\phi}_g(\mathbf{s}_i)\} \tag{4.11}$$

4.5 Adaptive Scale

Scale needs to be handled in our algorithm. Let us denote by σ_{prev} the size of object in the previous frame. For every position in the search process, we vary the scale by ± 10 percent, i.e., $\sigma = \sigma_{prev}$, $\sigma = \sigma_{prev} + \Delta\sigma$, $\sigma = \sigma_{prev} - \Delta\sigma$, where $\Delta\sigma = 0.1\sigma_{prev}$. Letting σ_{opt} denote the size of the best state determined by the algorithm, we avoid oversensitive scale adaptation by filtering the result, as described in [6]:

$$\sigma_{new} = \gamma\sigma_{opt} + (1 - \gamma)\sigma_{prev}, \tag{4.12}$$

where the default value of γ is 0.3.

Chapter 5

Experimental Results

In this chapter, we show the experimental results of our algorithm. In order to test the effectiveness and robustness of our approach, we run the algorithm on several sequences of a person, one obtained from [4] and the others captured in our lab. The sequences contain full 360-degree out-of-plane rotation of the target, cluttered backgrounds, and occlusion.

In our experiments, a person's head is modeled as a vertical ellipse with a fixed aspect ratio. The co-ordinates of the center of the ellipse is given by (x, y) and the length of the minor axis is σ. Then the state (location) of the head can be denoted by these parameters $\mathbf{s} = (x, y, \sigma)$. The tracking task is to find the best state $\mathbf{s}$ for every image frame. The state (location) of the head in the first frame is manually initialized.

5.1 Experiment on Cluttered Background

The video sequence used in Experiment 1 is obtained in our lab under a cluttered environment. The forward correlation search was run with a $\pm 10 \times \pm 10 \times \pm 1$ search window in x, y direction and scale. The backward correlation was run with a $\pm 5 \times \pm 5 \times \pm 1$ search window. Since the frame rate of the camera is high compared with the object's motion and backward correlation is applied for each frame (under textured background), we assume

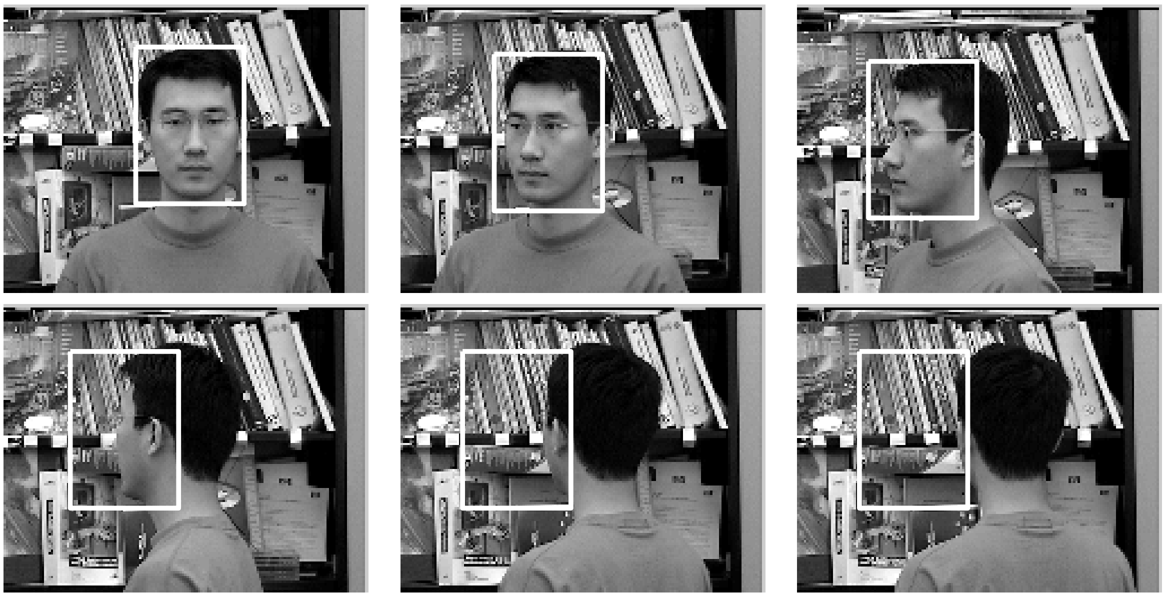


Figure 5.1: Tracking results of traditional template-based tracker on a sequence with cluttered background. Frames 10, 15, 20, 25, 28 and 30 are shown. The tracker drifts away from the head due to out-of-plane rotation and loses the target completely after several frames.

that no severe drift occurs for each frame. So here we use a smaller search window for backward correlation search to improve computational efficiency.

Figure (5.1) shows the tracking results of the traditional template-based tracker on this video sequence. Only the standard forward correlation search is used to track the person's head. The tracker drifts from the target when the object performs out-of-plane rotation. Because the tracker approximates the rotation with translation parallel to the image plane, it creates small errors in each image, which then accumulate to cause the tracker to lose the target completely after several frames.

Our algorithm was then run on the same video sequence. The tracking results are shown in Figure (5.2). In contrast, our tracker is able to successfully track the target throughout the sequence. Note in particular that the algorithm maintains a tight lock on the target in all the intermediate frames of the rotation. Figure (5.3) shows the tracking error of the above experiment.

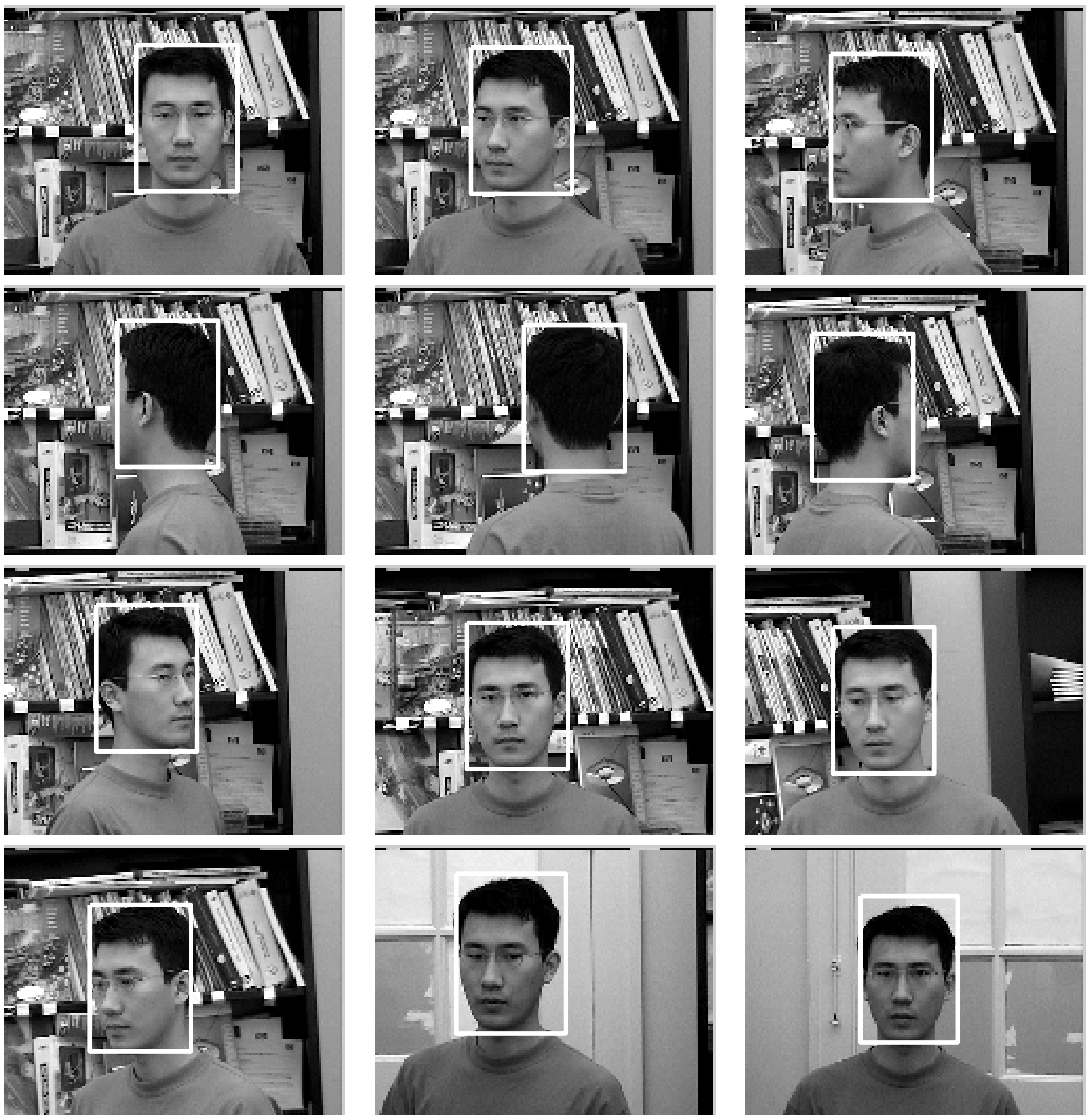
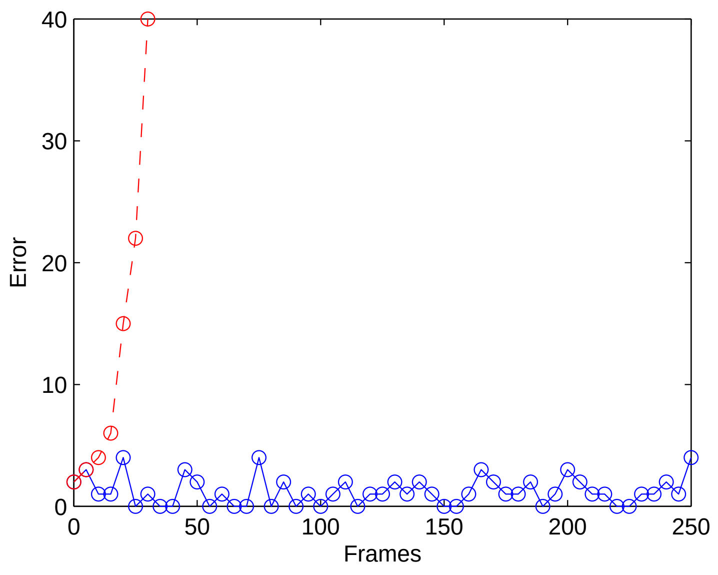
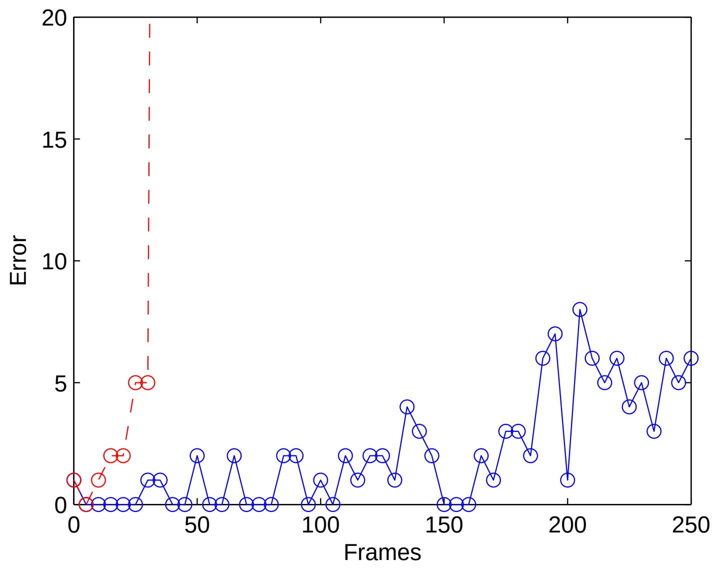


Figure 5.2: Tracking results of our algorithm on a sequence with cluttered background. Frame 10, 15, 20, 25, 30, 43, 51, 62, 164, 169, 197 and 239 are shown. The algorithm maintains a tight lock on the target in all the intermediate frames of the out-of-plane rotation.



(a)



(b)

Figure 5.3: Tracking error of Experiment 1 using our algorithm (solid) vs. the traditional template-based tracker (dashed). (a): tracking error in *x* direction. (b): tracking error in *y* direction.

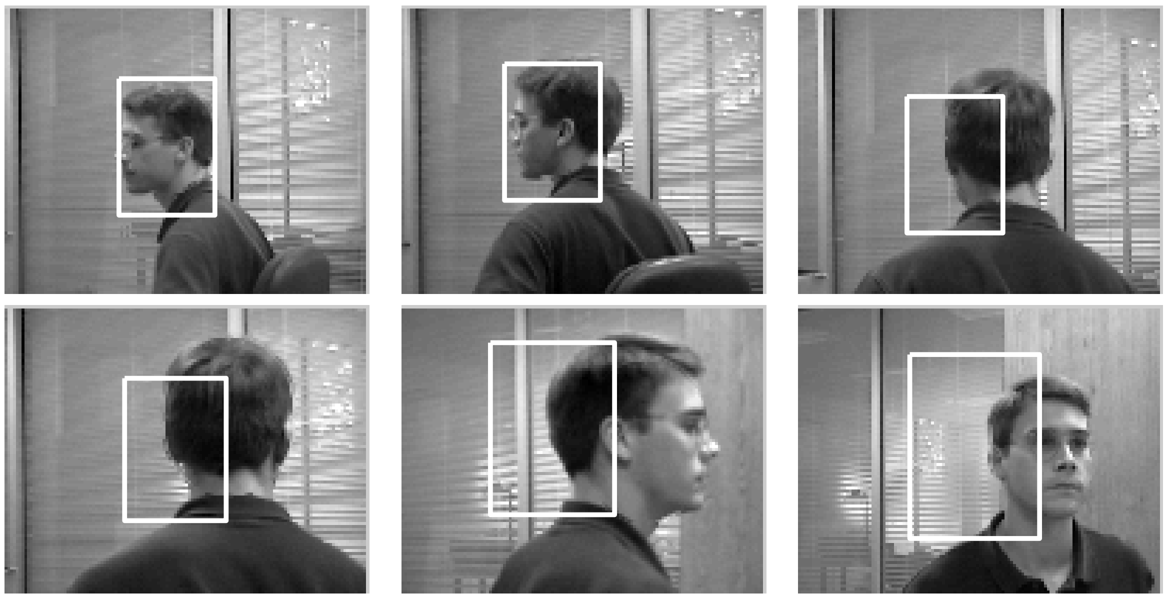


Figure 5.4: Tracking results of traditional template-based tracker on a sequence with untextured background. Frames 63, 67, 76, 81, 90 and 102 are shown. The tracker drifts away from the head due to out-of-plane rotation and loses the target completely after several frames.

5.2 Experiment on Untextured Background

In the second experiment, we tested our algorithm on a video sequence in which a man rotated and changed scale in front of an untextured background. We first ran the traditional template-based tracker on this sequence and the result is shown in Figure (5.4). We then ran our algorithm on the sequence and the result is shown in Figure (5.5). While the traditional template tracker drifts from the target, our algorithm is able to remain locked onto the target throughout all the intermediate frames of the sequence.

Figure (5.6) shows the likelihood of the backward correlation module at two snapshots from the first two experiments. When the background is textured, the backward correlation module produces a sharp peak near the true location, thus enabling the algorithm to remain on the target. However, when the background is untextured, there is not enough information in the image data for the backward correlation module to determine the correct location of the template in the previous frame. As a result, the gradient module is needed in order to avoid drift from the target.

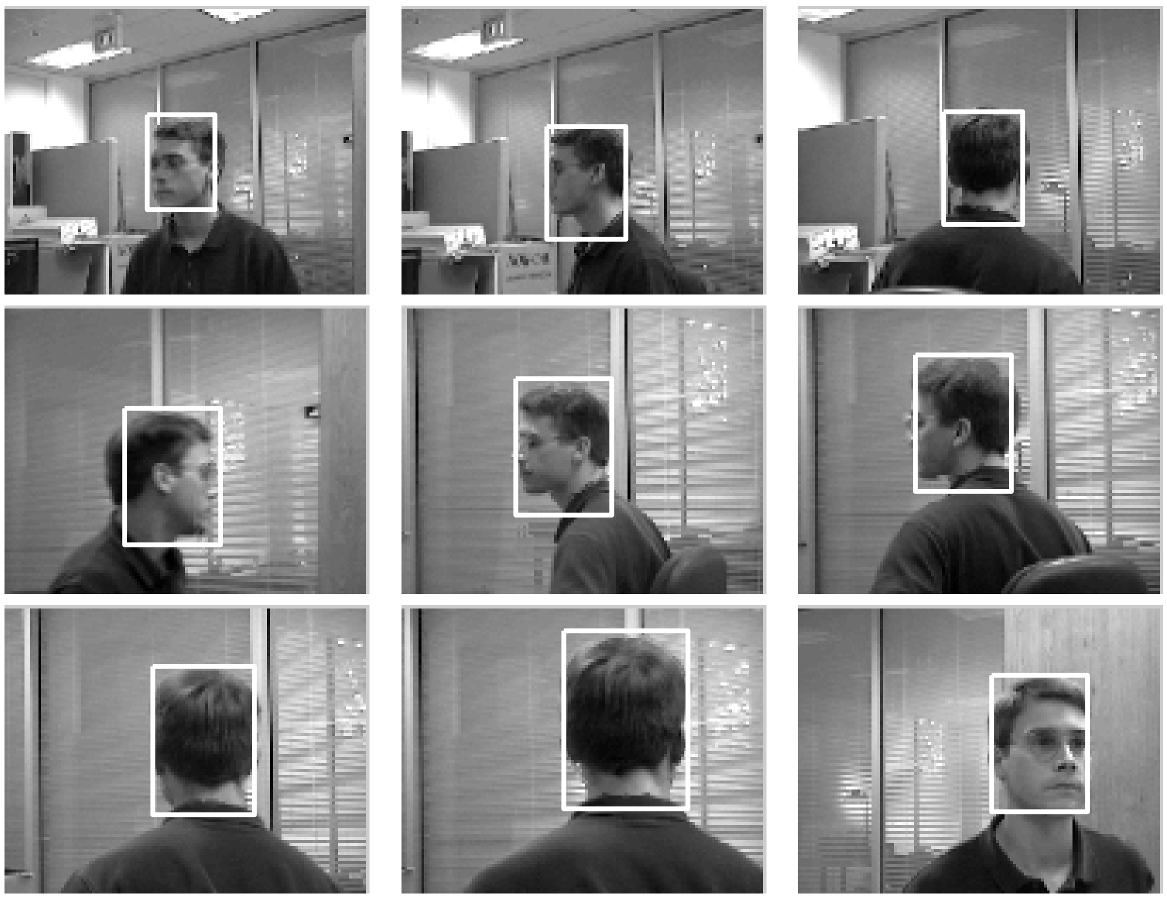
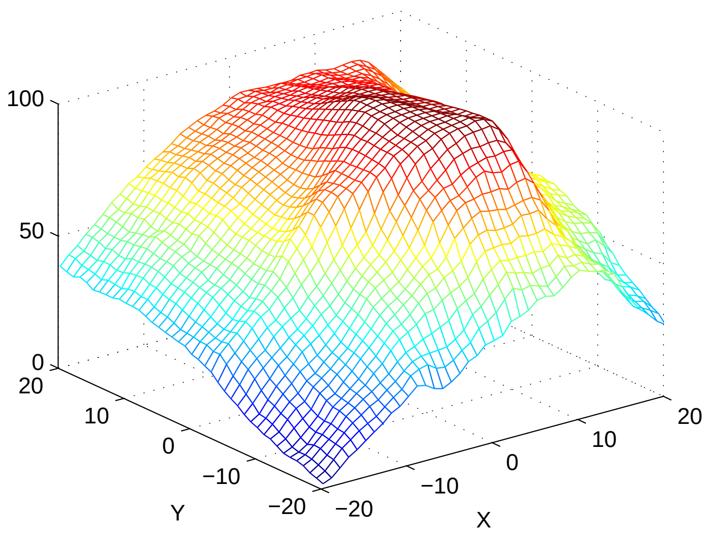
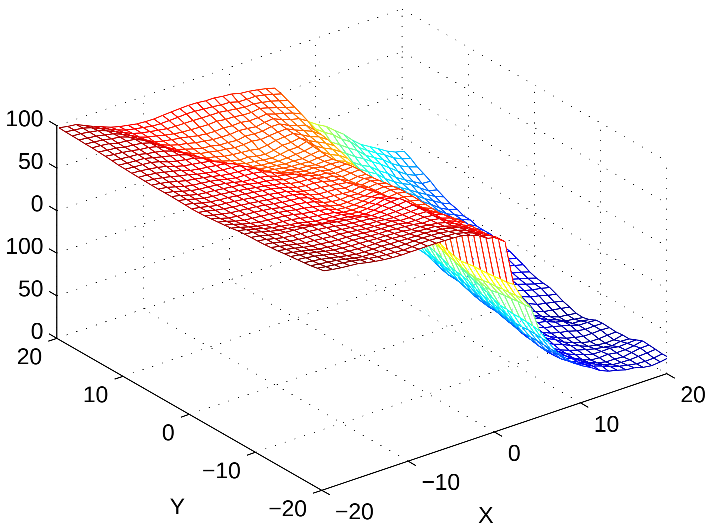


Figure 5.5: Tracking results of our algorithm on a sequence with untextured background. Frames 20, 25, 33, 44, 63, 67, 76, 81 and 102 are shown. The tracker is able to handle out-of-plane rotation, along with scale changes.

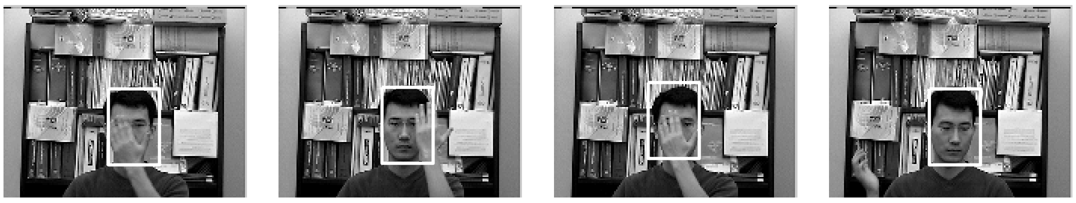


(a)

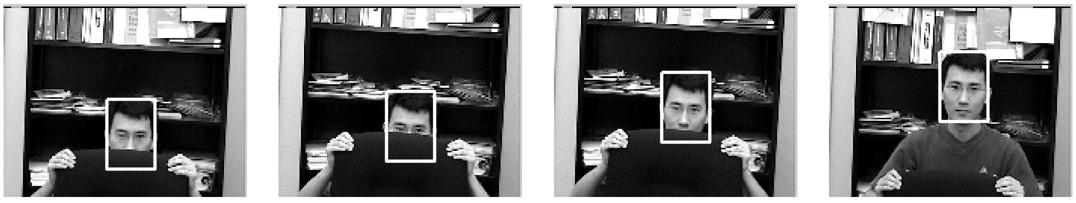


(b)

Figure 5.6: Likelihood of backward correlation module. (a): likelihood at a single scale for frame 25 of Experiment 1. (b): likelihood at a single scale for frame 63 of experiment 2. Backward correlation search starts at point $(0, 0)$. Frame 25 of Experiment 1 has a textured background around the head, and the peak of the likelihood gives the correct displacement. For frame 63 of experiment 2, the background around the head has little texture, and the likelihood is distracted by background pixels to the edge of the search range.



(a)



(b)

Figure 5.7: Tracking results on a sequence containing occlusion. (a): the head is partially occluded by the hand. (b): the head is partially occluded by a chair. Frames 34, 35, 40, 41, 55, 140, 143, 148, 152 and 156 are shown.

5.3 Other Experiments

Two experiments demonstrating the robustness of the proposed algorithm to moderate amounts of occlusion were conducted, and the tracking results are shown in Figure (5.7). We can see from Figure (5.7) that the tracker does not deviate, even when the target is occluded by up to 50%.

We also applied our algorithm to the problem of tracking a vehicle. In the video sequence, the SUV made a turn and was partially occluded by a pole. Figure (5.8) and Figure (5.9) show the tracking results using traditional template-based tracker and our algorithm, respectively. We can see from Figure (5.9) that, despite the significant pose changes in the vehicle, the tracker does not drift. Note also that our tracker is able to automatically adjust to the changing scale of the target.

Finally, we run our algorithm on a video sequence obtained from [4] which is shown in Figure (5.10). [4] failed on this sequence when the subject rotated because the histogram model had little hair. Despite the significant scale change during the out-of-plane rotation, our tracker is able to keep a lock on the target.

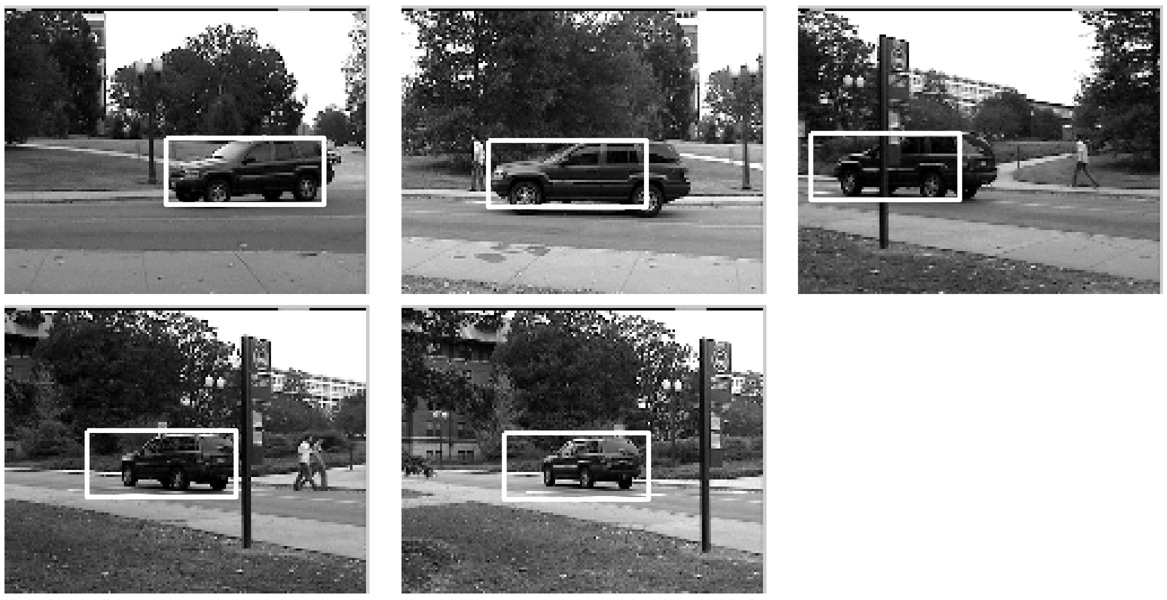


Figure 5.8: Tracking results of traditional template-based tracker on a sequence of a vehicle making a turn.

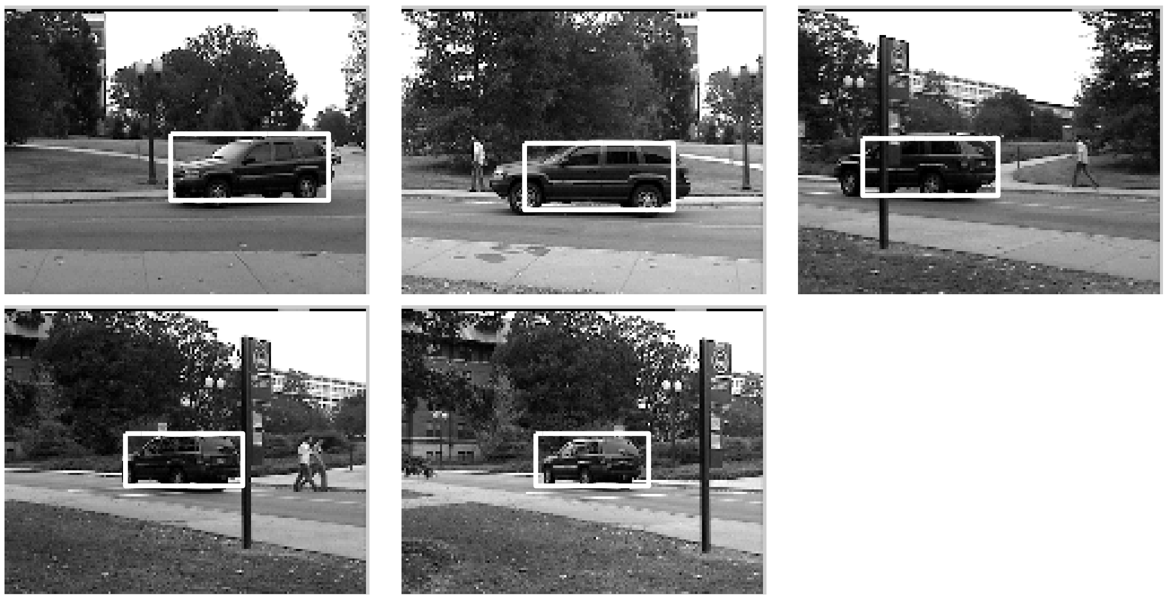


Figure 5.9: Tracking results of our algorithm on a sequence of a vehicle making a turn. The tracker is kept on the target despite of the significant pose changes of the vehicle. Scale is also well handled.

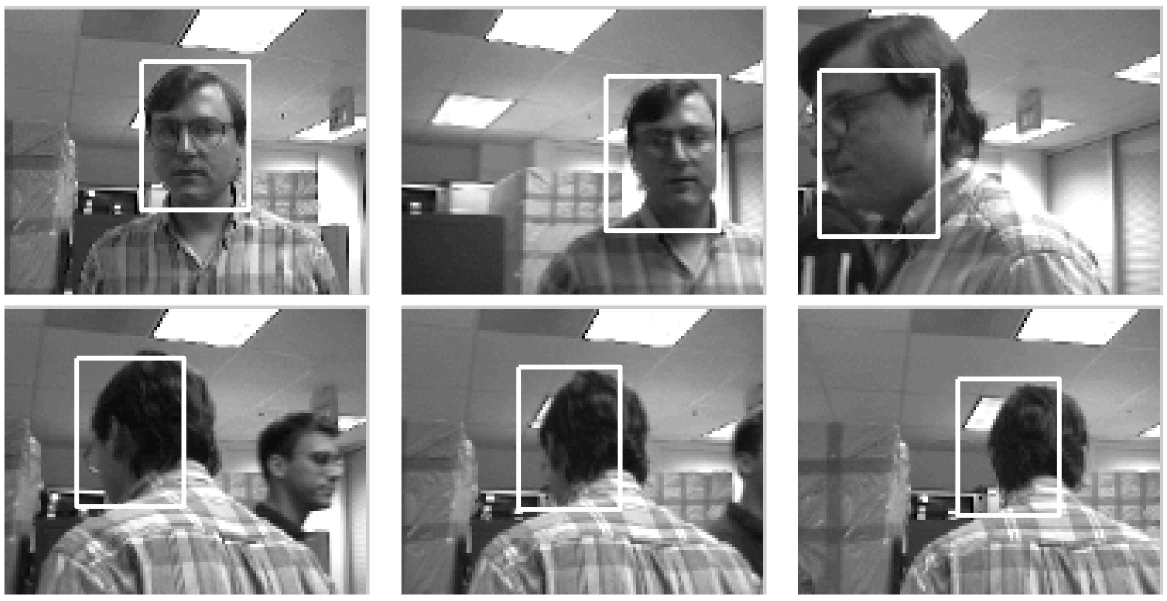


Figure 5.10: Tracking results of our algorithm on a sequence which histogram-based algorithm fails. Frames 2, 15. 22, 26, 27 and 28 are shown.

Chapter 6

Conclusion

We have presented an extension to a template-based tracker. By augmenting the standard forward correlation search with a backward correlation search, the algorithm achieves robustness to out-of-plane rotation, a problem which causes the traditional approach to fail. Such difficult changes in the appearance of the target have led many researchers to explore spatially-invariant features such as color histograms as an alternative to templates. Histograms, however, lack the specificity that is available with more detailed models such as templates. In this paper we have shown that it is possible, with very little computation, to overcome one of the fundamental limitations of template-based tracker.

The work presented in this thesis is only a beginning to explore the possibilities available to improve the performance of traditional template-based tracking. A natural extension to this work would be to use motion discontinuities around the perimeter of the object to further refine the description of the object's location. Motion vectors in the vicinity of the target would be an alternate way to guide the template to the correct location.

Bibliography

[1] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, http://www.ces.clemson.edu/˜stb/klt/.

[2] Shai Avidan. Ensemble tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[3] Stan Birchfield. An elliptical head tracker. In *Proceedings of the 31st Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1710–1714, 1997.

[4] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 232–237, 1998.

[5] Robert Collins, Yanxi Liu, and Marius Leordeanu. On-line selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631 – 1643, October 2005.

[6] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.

[7] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[8] Jeffrey Ho, Kuang-Chih Lee, Ming-Hsuan Yang, and David Kriegman. Visual tracking using learned subspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 782–789, 2004.

[9] Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 415–422, 2001.

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

[11] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis and Machine Vision*. Thomson Learning, 2002.

[12] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[13] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[14] Kentaro Toyama and Gregory D. Hager. Incremental focus of attention for robust visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 189–195, 1996.

[15] Yaser Yacoob and Larry S. Davis. Tracking rigid motion using a compact-structure constraint. In *ICCV (1)*, pages 198–205, 1999.