A NOVEL ZOOM INVARIANT VIDEO OBJECT TRACKING ALGORITHM (ZIVOTA)

by

Yankun Wei

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA, CANADA

June 2003

© Yankun Wei 2003
ABSTRACT

Advanced surveillance system should have the ability to track targets that have non-rigid shape and structure/size. A limitation in the field of real-time target tracking is the high computation load, so the tracking algorithms are difficult to be implemented on an ordinary personal computer without any special-purpose hardware component.

This thesis presents a real-time zoom-invariant video object tracking algorithm (ZIVOTA). It can be used to track human or rigid-shaped object. ZIVOTA can provide information of object-of-interest while performing tracking, which enables automatic visual system control. It employs the same set of features to track both the position and size of a moving object through frames. Time-consuming processes like feature detection and affine basis set selection will be done only when it is necessary. Computational load is largely reduced. By using affine motion model, this technique is fundamentally zoom-invariant, it can deal with translation, rotation, and scaling of objects. Since the gaze point (representing the object location) and object boundary points (representing the object size) are virtual, ZIVOTA is able to handle partial occlusion.
ACKNOWLEDGEMENT

I would like to express my sincere appreciation to my supervisor Dr. Wael Badawy for providing me the opportunity to work on this exciting project. His constant support, encouragement, patience and invaluable guidance are highly appreciated. I gratefully acknowledge the Department of Electrical and Computer Engineering, University of Calgary and Olympic Oval for their financial support.
TABLE OF CONTENTS

Approval Page....................................................................................... ii
Abstract................................................................................................ iii
Acknowledgement.................................................................................. .iv
Table of Contents..................................................................................... v
List of Figures.......................................................................................... viii

**Chapter 1. Introduction ................................................................. 1**

1.1 Active Video Object Tracking..................................................... 1
1.2 Research Motivation And Objective............................................. 3
1.3 Thesis Outline.............................................................................. 6

**Chapter 2. Video Object Detection and Tracking ............................ 7**

2.1 Still Image Segmentation Algorithms........................................... 8

2.1.1 Histogram thresholding............................................................ 8
2.1.2 Watershed segmentation.......................................................... 9
2.1.3 Edge detection......................................................................... 10
2.1.4 Active contour model.............................................................. 12
2.1.5 Corner detection..................................................................... 14

2.1.5.1 The Curvature Scale Space (CSS) Corner Detector............. 15
2.1.5.2 The Plessey Feature Point Detector................................. 15
2.1.5.3 The SUSAN corner detector........................................... 16
2.1.6 Region growing..................................................................... 17

2.2 Motion-Based Segmentation....................................................... 18
4.2.2 "speed skater" sequence ................................................................. 63
4.3 Real-time Experimental Results ....................................................... 67
4.4 Performance Analysis ..................................................................... 70
  4.4.1 Computational Cost .................................................................. 71
  4.4.2 Measuring Confidence .............................................................. 73

Chapter 5. Conclusion and Future Work ........................................... 80

  5.1 Summary .................................................................................... 80
  5.2 Discussion And Future Work ...................................................... 81
LIST OF FIGURES

Figure 1.1. The proposed real-time tracking system .......................................................... 4

Figure 2.1. Example of histogram thresholding (A shallow valley point exists implying
that a threshold could be used in discriminating the object from the background) .... 9

Figure 2.2. Basic theory of watershed ........................................................................ 10

Figure 2.3. Frame difference of the car sequence ......................................................... 20

Figure 2.4. Optical flow of the moving car .................................................................. 22

Figure 2.5. Block based object segmentation .............................................................. 23

Figure 2.6. HASM mesh topology .............................................................................. 25

Figure 2.7. Mesh topology after merge and split ......................................................... 26

Figure 3.1. Example of masks used in SUSAN feature detection .............................. 33

Figure 3.2. SUSAN masks with similarity colouring .................................................... 33

Figure 3.3. Digital approximation of SUSAN mask ..................................................... 34

Figure 3.4. Output of SUSAN “corner” detector ......................................................... 38

Figure 3.5. Affine transformation example ................................................................. 40

Figure 3.6. A triangle distortion ................................................................................. 42

Figure 3.7. Object boundary box detection. The gray blocks are changed blocks and they
form a moving object. The boundary box covers all the changed blocks ................. 43

Figure 3.8. Object boundary points and gaze point example. The cross is the object
boundary center and also the gaze point, the 4 circles are the object boundary points. ................................. 43
Figure 3. 9. Future points in object boundary box, the first picture gives the object boundary box, the second picture shows the detected corners within the object boundary box. ................................................................. 45

Figure 3. 10. Selecting different affine basis set from feature points. ......................... 46

Figure 3. 11. Example of head selection. (a). The original picture. (b). The frame difference result. (c). Morphological filtering output. (d). Vertical projection of. 48

Figure 3. 12. Motion estimation of affine basis points. .................................................. 49

Figure 3.13. Forward correspondence estimation.................................................................. 51

Figure 3.14. Backward correspondence estimation............................................................ 51

Figure 3. 15. Affine object tracking ................................................................................. 52

Figure 3. 16. The change of affine basis set .................................................................. 54

Figure 4. 1. Prototype system ......................................................................................... 55

Figure 4. 2. The graphic user interface ............................................................................. 56

Figure 4. 3. Example of camera position control................................................................. 58

Figure 4. 4. Video signal processing block diagram............................................................ 59

Figure 4. 5. The “robot” sequence showing a toy car moving towards the camera........ 61

Figure 4. 6. Moving object detection on “car” sequence.................................................... 61

Figure 4. 7. Feature points detection result ....................................................................... 61

Figure 4. 8. Affine basis points selected from feature points ............................................ 62

Figure 4. 9. ZIVOTA tracking result of “car” sequence..................................................... 62

Figure 4. 10. The skater sequence ................................................................................... 63

Figure 4. 11. Moving object detection on “skater” sequence ............................................ 64

Figure 4. 12. Frame difference ....................................................................................... 65
Figure 4. 13. Head position detection using projection. (a) Morphological filtering output of foreground region, (b) detected moving object, (c) vertical projection histogram.

Figure 4. 14. Affine basis selection on “speed skater” sequence.

Figure 4. 15. ZIVOTA tracking result of “speed skater” sequence.

Figure 4. 16. Real-time experiment result captured through the lens of active camera.

Figure 4. 17. Zooming while tracking.

Figure 4. 18. Tracking while there is partial occlusion.

Figure 4. 19. ZIVOTA tracking results of a robot moving from right to left (sequence 1).

Figure 4. 20. Analysis of processing time.

Figure 4. 21. Demo of predicted object trajectory and size.

Figure 4. 22. ZIVOTA predicted trajectory of the gaze point (the video sequence used is the same as Figure 4. 19, sequence 1).

Figure 4. 23. ZIVOTA predicted size of the moving object (sequence 1).

Figure 4. 24. The size prediction confidence of ZIVOTA (sequence 1).

Figure 4. 25. The position prediction confidence of ZIVOTA (sequence 1).

Figure 4. 26. The measurement confidence of ZIVOTA (sequence 1).
CHAPTER 1

INTRODUCTION

In recent years, the increasing requirements for real time surveillance and target tracking have emerged the research in advanced digital video signal processing. The availability of high performance video sensors and processor speed make it possible to develop an integrated real time object tracking system.

1.1 Active Video Object Tracking

A basic requirement of active tracking system is the ability to fixate or track video objects using an active camera. Real-time object tracking systems have been developed recently. T. Darrel’s system [1] combines stereo, color, and face detection modules into a single robust system. Dense stereo processing is used to isolate humans, skin-hue classification identifies and tracks likely body parts. Face pattern detection discriminates and localizes the face. The system is based on three primary visual processing modules: depth estimation, color segmentation, and intensity pattern classification. It is implemented on a card using multi-FPGA (Field Programmable Gate Array) computing engine.

Pfinder (person finder) [2] uses a multi-class statistical model of color and shape to obtain a 2-D representation of head and hands in a wide range of viewing conditions. The initialization procedure obtains descriptions of the person from weak prior knowledge about the scene and is necessary for bootstrapping the system at the start and when tracking breaks down. The tracking procedure recursively updates the description based
on strong prior knowledge from the previous frame. The tracking procedure can determine when it is in error and can then defer to initialization procedure, which is slower but more reliable. Initialization and tracking procedures for pfinder is based largely on a Maximum A Posteriori probability (MAP) approach. Spfinder is a recent extension of Pfinder in which a wide-baseline stereo camera is used to obtain 3-D models. Spfinder has been used in a smaller desk-area environment to capture accurate 3D movements of head and hands. Its real-time performance could not be achieved without special-purpose hardware to speed up the processing.

KidRooms [3] is a tracking system based on closed-world regions, where regions of space and time in which the specific context of what is in the regions is assumed to be known. These regions are tracked in real-time domains where object motions are not smooth or rigid, and where multiple objects are interacting. Multivariate Gaussian models are applied to find the most likely match of human subjects between consecutive frames taken by cameras mounted in various locations. Bregler [3] uses many levels of representation based on mixture models, EM, and recursive Kalman and Markov estimation to learn and recognize human dynamics.

Lipton’s system [4] extracts moving targets from a real-time video stream, classifies them into pre-defined categories and tracks them. Because it uses correlation matching, it is primarily targeted at the tracking of rigid objects.

Birchfield [5] proposed an algorithm for tracking a person’s head by modeling the head as an ellipse whose position and size are continually updated by a local search combining the output of a module concentrating on the intensity gradient around the
ellipse's perimeter, and another module focusing on the color histogram of the ellipse's interior.

Reid and Murry [7][8] introduced monocular fixation using affine transfer as a way of a cluster of such features, while at the same time respecting the transient nature of the individual features. By using transfer to move the fixation point, the method was able to exploit the robustness arising from rigid scene structure without requiring a prior model. It needed only one camera. It was a promising process for tracking while zooming.

Hayman et al. [9] propose a zooming while tracking system that performs only monocularly and it was a struggle to obtain sufficient data. Their system only deals with the transfer of fixation/gaze point, and it provides no information regarding the structure or size of objects. Moreover, it requires manually setting of object-of-interest.

1.2 Research Motivation And Objective

Advanced surveillance system should have the ability to track targets that have non-rigid shape and structure/size. A limitation in the field of real-time target tracking is the high computation load, so the tracking algorithms are difficult to be implemented on an ordinary personal computer without any special-purpose hardware component.

The proposed research is motivated by considerations of a tracking system that monitors the movements of a speed skater. The tracking system must detect the moving skater and identify his position and size to keep him inside the captured video frame with sufficient resolution. The captured video sequences will be used by the coaches to analyze the performance of the skaters. The coaches can be released from standing there holding a camcorder and manually doing the adjustment. A computer controlled video
camera and one personal computer are used to deal with all the tasks, as in Figure 3.1. This application brings up the question of whether there is any advanced video image processing algorithms that satisfies all our requirements. After an extensive study in this field, no similar tracking system can be found. The moving skater has a non-rigid shape, tracking algorithms dealing with changing–shape objects are computational intensive.

Figure 3.1.1. The proposed real-time tracking system

The primary objective of this research is to develop a low-cost real-time object-tracking algorithm, that is termed Zoom-Invariant Video Object Tracking Algorithm (ZIVOTA), which could be implemented on an ordinary personal computer to perform real-time surveillance.
ZIVOTA can track not only the trajectory but also the structure of the non-rigid shaped moving objects. It constructs dynamic motion models of the target during the process of tracking. ZIVOTA employs a combination of image analysis techniques for tracking to detect and track objects, even when they are partially occluded. ZIVOTA can deal with translation, rotation and scaling, it is fundamentally invariant to zoom. In order to reduce the computational cost, only a set of feature points of a target will be tracked to predict the object location and size.

The proposed ZIVOTA algorithm is prototyped using an active camera system, a camera switch, a video capture card and a personal computer which has a 1.8GHz Pentium processor. It can control zoom, pan, and tilt of the camera to achieve sufficient resolution to analyze the target’s actions.

The major features of ZIVOTA are as follows:

- Utilizes the advantages of both still image processing and motion-based image segmentation to save computation
- Constructs a zoom-invariant motion model
- Tracks the position and size of object
- Tracks both rigid objects like cars and non-rigid objects like human
- Controls an active camera to track moving objects

The potential application areas include:

- Coaching tools: ZIVOTA automatically tracks the athlete and captures video sequences for analysis.
• Security Surveillance: ZIVOTA can be used in an automatic system that has the ability to extract information from video streams and provides warnings to human observers.

• Biomedical image processing: Cells are also moving objects that can be tracked by ZIVOTA.

• Videoconference and education: By using ZIVOTA, the users will no longer need to sit straightly before a camera, they can walk around the room with the camera always following them.

• Child Care: ZIVOTA can detect and track children and enables their parents to keep an eye on the child while they are working in another zoom.

1.3 Thesis Outline

This thesis is organized as follows. A literature review of object detection and tracking is given in chapter 2. Chapter 3 describes the proposed real-time zoom-invariant video object tracking algorithm (ZIVOTA). Chapter 4 describes the prototype system and implementation. Tracking results and performance analysis are also given in detail. Chapter 5 discusses the results and concludes this work, and future work is suggested.
Chapter 2

VIDEO OBJECT DETECTION AND TRACKING

A tracking system should have the ability to control an active camera to follow a moving target. The camera actions are dependent to the moving target, so the detection and measurement of video object are crucial to the automatic tracking system. Image segmentation techniques are employed and the moving object and its feature are extracted respectively. Then the extracted object is tracked, its position and size information will be used to control the camera. In this chapter, some image segmentation and tracking algorithms are introduced.

Video based object detection and tracking [10] is a process that identifies moving object, and then following that object as it moves through a sequence of consequent images. A sequence of images is segmented into a set of arbitrarily shaped regions where each of the regions may represent a particular content of the video stream.

Segmentation can be used for object detection and identification. It recognizes homogeneous regions within an image as distinct and belonging to different objects. The segmentation process can be based on finding the maximum homogeneity in gray levels or in motion information within the regions identified. In general, two approaches are used for the object segmentation [16][17]. The first is still image-based segmentation, in which an initial segmentation is performed on the first image frame and the object is extracted. Then the instants of the object are searched in the consequent image frames.
The other approach is motion-based segmentation, in which an object is segmented in each and every image frame. The segmentation technique is used in all frames of the video sequence. Motion cues like motion vectors are used to link extracted foreground areas together.

In general, the cues that can be used for object segmentation are

- Low level features like color, texture, and shape;
- Similarities in motion vectors of adjacent regions;
- Semantic information between parts of objects;
- Context specific properties of objects like size, position, etc.

2.1 Still Image Segmentation Algorithms

Still image segmentation consists of two main approaches:

(1) locating boundaries or edges of regions, for e.g. edge detection [18], active contour [19].

(2) grouping points into homogeneous regions, which then permits determination of the respective boundaries, for e.g. histogram thresholding [22], watershed [23].

2.1.1 Histogram thresholding

The histogram thresholding technique was originally proposed by Ohlander [17], and then extended by Cheriet et al [22]. The thresholding operation partitions pixels of an image into two classes: (1) object and (2) background. It is based on the constructs of the color and hue histograms. The image, as in Figure 2.1, is thresholded at its separated peaks. The minima between these peaks represent optical density values with low
frequency of occurrence and should correspond to the boundaries between parts of the image.

The advantage of this method is that it can be developed without any constraint on the number of objects in the digital image. This approach is however only suitably applied when the target object is the darkest or the lightest object in a given image and fails when the histogram is unimodal. Moreover, finding the optimal threshold to decompose the images is a very time-consuming task.

Figure 2.1. Example of histogram thresholding (A shallow valley point exists implying that a threshold could be used in discriminating the object from the background)

2.1.2 Watershed segmentation

The introduction of the watershed transformation as a morphological tool was done by H. Digabel Later, S. Beucher et al. [25] analyzed watersheds from an immersion analogy perspective. The idea of watershed is realized by considering an image as a topographic
surface. The image intensity (the gray level) is considered as an altitude with this point of view. For the image segmentation purpose, the watershed algorithm is usually applied on the gradient image. In the gradient image, a homogeneous region becomes the inner region of a catchment basin and the region boundary becomes the watershed, as shown in Figure 2.2. To characterize object boundaries, we look for crest lines of the gradient image located on the watersheds.

The advantage of the watershed method is that it generates a closed boundary. Its disadvantage is that it can easily cause an over-segmentation or under-segmentation problem.

2.1.3 Edge detection

One-dimensional features such as edges and lines in images are also used for object segmentation[17]. Edge detection algorithms produce new images that show higher intensity in pixels near gray value transitions. The edge operator is a neighborhood operation that determines the extent to which each pixel's neighborhood can be partitioned by a simple arc passing through the pixel where pixels in the neighborhood on
one side of the arc have one predominant value and pixels in the neighborhood on the
other side of the arc have a different predominant value.

An edge pixel is described using two important features: (1) Edge strength, which is
equal to the magnitude of the gradient; (2) Edge direction, which is equal to the angle of
the gradient.

Some widely used edge detection operators are introduced.

(1) Roberts Cross Operator

The Roberts cross operator provides a simple approximation to the magnitude of the
gradient $G[f[i,j]]$, where

$$G[f[i,j]] = |f[i,j] - f[i+1,j+1]| + |f[i+1,j] - f[i,j+1]|$$  (2.6)

where $f[i,j]$ denotes image pixel intensity, and $[i,j]$ is the position of the pixel.

Using a substitute in convolution masks, equation (2.6) becomes:

$$G[f[i,j]] = |G_x| + |G_y|$$  (2.7)

where $G_x$ and $G_y$ are calculated as:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$  (2.8)

(2) Sobel operator

Normally, it uses a $3 \times 3$ neighborhood for gradient calculations.

For a pixel $f[i,j]$, the Sobel operator is the magnitude of the gradient computed by

$$\sqrt{s_x^2 + s_y^2},$$

where $s_x$ and $s_y$ can be implemented using the following convolution

masks:
$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ \hfill (2.9)

$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ \hfill (2.10)

$s_x$ is simply $s_y$ rotated by 90°. This operator places greater emphasis on pixels that are closer to the center of the mask.

(3) Prewitt operator

The Prewitt operator uses the same equations as the Sobel operator except that $s_x$ and $s_y$ are different, as given below.

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$ \hfill (2.11)

$$s_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$ \hfill (2.12)

Unlike the Sobel operator, Prewitt operator does not place any emphasis on pixels that are closer to the center of the masks.

2.1.4 Active contour model

Active contour models were introduced by Kass et al [19] as a powerful solution to the low-level imaging task of finding salient contours (e.g. edges and lines) in digitized
images. An interesting and powerful property of an active contour model is its ability to find subjective contours and interpolate across gaps in edge chains.

The active contour model, also called snake, is an energy-minimizing spline influenced by external constraint forces and internal smoothness constraint forces that is guided by image forces that pull the contour toward image features such as edges. In other words, besides being controlled by the edges (image forces), the snake contour can be manipulated and constrained by some other forces, such as a priori knowledge about the contour (external forces) and smoothness and sharpness constraints about the contour (internal forces). The snake paradigm models a deformable contour possessing image energy, external energy and internal energy. When the contour is acted on by these three energy fields, the contour seeks equilibrium at a minimum of total energy field by moving and changing shape. That is how the technique gets the name “snake”.

A contour is represented parametrically as $V(s) = (V_x(s), V_y(s))$, where $V_x$ and $V_y$ are the $x$ and $y$ coordinates respectively of any point on the contour and $s$ is the parameter. Conventionally the value of $s$ is chosen to change from 0 to 1 when the contour goes from the beginning to the end. The energy of the contour is defined as:

$$ E_{snake-total} = \int E_{snake}(V(s))ds $$

$$ = \int (E_{int}(V(s)) + E_{ext}(V(s)) + E_{img}(V(s)))ds \quad (2.13) $$

where $E_{int}$ represents the internal force (or smoothness constraints) which has both stretching and bending terms, $E_{img}$ stands for the image force constructed from some
image features to attract the snake to the edges, $E_{ext}$ is the force due to extraneous constraints which can be derived from knowledge bases and/or users.

The strategy of snakes is to deform the contour to the locations that minimizes the total force. A contour that fits the image features, a priori knowledge and smoothness constraints is obtained by balancing the internal forces, the external forces, and the image forces.

### 2.1.5 Corner detection

Corner detection is an approach of image processing that detects two dimensional image features, including corners junctions, etc. Image corner detection is an important task in various computer vision and image understanding systems. Applications include motion tracking, object recognition, and stereo matching.

A few methods use binary edge maps to find corners. The edges are detected and then edge curvature is calculated in order to find corner locations [27][28][29].

Some methods find points of localized image structure directly[30].

Corner detection should satisfy a number of important criterions:

- All the true corners should be detected.
- No false corners should be detected.
- Corner points should be well localized.
- Corner detector should be robust with respect to noise.
- Corner detector should be efficient.
2.1.5.1 The Curvature Scale Space (CSS) Corner Detector

The CSS corner detector [27] works as follows:

- Extract the edge contours from the input image using any suitable edge detector.
- Fill small gaps in edge contours. When the gap forms a T-junction, mark it as a T-corner.
- Compute curvature on the edge contours at a high scale.
- The corner points are defined as the maxima of absolute curvature that are above a threshold value.
- Track the corners through multiple lower scales to improve localization.
- Compare T-corners to the corners found using the CSS procedure and remove very close corners.

Experimental results show this algorithm spends most of its time (80%) detecting the edges in the image. Faster edge detectors may be used. The local maxima of absolute curvature are the possible candidates for corner points. A local maximum is either a corner, the top value of a rounded corner or a peak due to noise. The latter two should not be detected as corners. The curvature of a real corner point has a higher value than that of a rounded corner or noise. The corner points are also compared to the two neighboring local minima. The curvature of a corner should be twice that of one of the neighboring local minima. This is because when the shape of the contour is very round, contour curvature values can be above the threshold, and false corners may be declared.

2.1.5.2 The Plessey Feature Point Detector
In [30] Harris and Stephens described what has become known as the Plessey feature point detector. We can see the outline of how it works if we consider the following matrix:

\[
M = \begin{bmatrix}
\left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\
\left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2
\end{bmatrix}
\] (2.14)

where \(I(x; y)\) is the grey level intensity of the image. If at a certain point the two eigenvalues of the matrix \(M\) are large, then a small motion in any direction will cause an important change of grey level. This indicates that the point is a corner.

Corners are defined as local maxima of the corner response function. Sub-pixel precision is achieved through a quadratic approximation of the neighborhood of the local maxima. To avoid corners due to image noise, it can be useful to smooth the images with a filter first.

In practice too much corners are often extracted. In this case it is helpful to first restrict the numbers of corners before trying to match them. One possibility consists of only selecting the corners with a value \(R\) above a certain threshold. This threshold can be tuned to yield the desired number of features. Since for some scenes most of the strongest corners are located in the same area, it can be interesting to refine this scheme further to ensure that in every part of the image a sufficient number of corners are found.

### 2.1.5.3 The SUSAN corner detector

SUSAN (Smallest Univalued Segment Assimilating Nucleus) [31] presents an
entirely different approach to low level image processing compared to all pre-existing algorithms. It provides corner detection as well as edge detection and is more resistant to image noise and no noise reduction is needed.

This approach has been chosen to be the feature detector in our hybrid tracking algorithm. Its details will be given in chapter 3.

2.1.6 Region growing

Region growing algorithms [31] take one or more pixels, called seeds, and grow regions around them based upon certain homogeneity criteria. If the adjoining pixels are similar to the seed, they are merged into a single region. The process continues until all the pixels in the image are assigned to one or more regions. For region growing, seeds can be automatically or manually selected. The automated selection can be based on finding pixels that are of interest, e.g. the brightest pixel in an image can serve as a seed pixel. They can also be determined from the peaks found in an image histogram. On the other hand, seeds can also be selected manually for every object presented in the image.

Although many still image segmentation algorithms can be used in the detection of the moving object, we still incline to the motion-based segmentation for the following reasons:

- Still image segmentation algorithms need too much prior knowledge or constraints, such as the semi-manual active contour method.
- The segmented regions are not semantically meaningful.
- The processing is for both moving and stationary objects. So it is very time-consuming.
• For the segmentation of video sequences, the object segmentation has very repeated computational tasks.

2.2 MOTION-BASED SEGMENTATION

Motion-based segmentation is different from still image segmentation in that motion is the most important feature for segmentation. A motion based algorithm for object segmentation starts by detecting changes in image frames. It depends on the gray value differences between the frames, which are highly susceptible to both noise and/or drifts in luminance sources. The local sum of absolute differences is compared against a threshold for identifying changed areas. Object motion tracking includes first describing where there is motion and then describing the character of that motion. Detection of moving objects in video streams is the first task to be performed in automated motion tracking system.

Aside from the intrinsic usefulness of being able to segment video streams into moving and background components, detecting moving objects provides a focus of attention for recognition, classification, and activity analysis, making these later processes more efficient since only “foreground” pixels need to be considered.

To build a temporal model of activity, individual objects generated by motion detection are tracked over time by matching them between frames in the video sequence. Given a moving object region in the current frame, we determine the best match in the next frame by performing image correlation matching, computed by convolving the object’s intensity template over candidate regions in the new image. The commonly used
motion-based segmentation algorithms involve temporal difference and dense optical flow fields. In the following, we will describe them at length.

### 2.2.1 Temporal difference

A traditionally used segmentation method is using a threshold to segment a video frame into "changed" versus "unchanged" regions with respect to the previous frame [15]. The unchanged regions denote the stationary background, while the changed regions denote the moving and occlusion areas. The frame difference $TD_{k,k-1}(x_1,x_2)$ is defined between the frames $k$ and $k-1$ as

$$TD_{k,k-1}(x_1,x_2) = s(x_1,x_2,k) - s(x_1,x_2,k-1) \quad (2.15)$$

which is the pixel-by-pixel difference between the two frames. Assuming that the illumination remains more or less constant from frame to frame, the pixel locations where $TD_{k,k-1}(x_1,x_2)$ differs from zero indicate "changed" regions. However, the frame difference hardly ever becomes exactly zero, because of the presence of observation noise.

In order to distinguish the nonzero differences that are due to noise from those that are due to a scene change, segmentation can be achieved by thresholding the difference images as

$$z_{k,k-1}(x_1,x_2) = \begin{cases} 1 & \text{if } |FD_{k,k-1}(x_1,x_2)| > T \\ 0 & \text{otherwise} \end{cases} \quad (2.16)$$
where $T$ is an appropriate threshold. Here, $z_{k,k-1}(x_1, x_2)$ is called as segmentation label field, which is equal to "1" for changed regions and "0" otherwise. An example of this approach is shown below:

![Frame difference](image)

**Figure 2. 3. Frame difference of the car sequence**

### 2.2.2 Optical flow methods

Optical flow [10] estimation is frequently used for segmentation and recognition of moving objects. Optical flow is the velocity field which warps one image into another (usually very similar) image.

The fundamental assumption enabling optical flow estimation is brightness conservation. It is based on the estimation of a measure of the change of image brightness in the frame sequence to find the velocity field.
where \( \nu = (v_1, v_2) \) is the optical flow vector. An optical flow vector is defined as the temporal rate of change of the image-plane coordinates, at a particular point \((x, t)\). It corresponds to the instantaneous pixel velocity vector.

\[
\nu(x_1, x_2) = \left( \frac{dx_1}{dt}, \frac{dx_2}{dt} \right)
\]

(2.18)

Optical Flow Constraint Equation suppose that the image intensity is given by \( I(x, y, t) \), where the intensity is now a function of time, \( t \), as well as of \( x \) and \( y \). We get the Optical flow equation, which is the basic of various optical flow based methods.

\[
\begin{bmatrix}
\frac{\partial I}{\partial x_1} & \frac{\partial I}{\partial x_2}
\end{bmatrix}^T \begin{bmatrix}
\nu_1 \\
\nu_2
\end{bmatrix} = -\frac{\partial I}{\partial t}
\]

(2.19)

After calculating the optical flow vectors of every pixel within the frame, a threshold could be used to filter out the background noise. Pixels with the vector higher than the threshold and have similar flow vectors are determined to be the target object. In practice, the optical flow field is defined as a vector field of pixel velocities on a spatial-temporal lattice.

As in Figure 2.4. The car is moving from right to left, generating the optical flow field shown in the center.
2.2.3 Block-based method

Block-based methods [10] assume that the image is composed of moving blocks. Each block is assigned with only one motion vector. This method is somewhat similar to optical flow method except that the motion vectors are calculated on the block level other than pixel level. It could be described as an N*N block centered about the pixel \( n = (n_1, n_2) \), which is modeled as a globally shifted version of a same-sized block in frame \( k+l \).

\[
b(n_1, n_2, k) = b(n_1 + d_1, n_2 + d_2, k + 1) \quad (2.20)
\]

To extract the moving object, the blocks with the similar motion vectors could be connected together to represent the target. It is shown in the following page.
Compared with the optical flow method, the computation of motion vector is largely reduced, but it suffers from the selection of the block size. A large block size results in less computation but with a lack of accuracy of motion detection, while a small size may add more redundancy.

### 2.2.4 Mesh-based method

A 2-D mesh is a planar graph that partitions a 2-D image region into polygonal patches. The vertices of the polygonal patches are referred to as node points. Mostly, the patches are triangles of quadrangles, leading to triangular or quadrilateral meshes respectively. The mesh-based motion estimation algorithm generates the mesh and estimates the motion vectors of the mesh nodes. The mesh-based motion representation can be used for both video compression and object tracking.
There are different mesh structures which can be used for motion estimation and segmentation. They are Uniform mesh, Delaunay mesh [37], Hierarchical Adaptive Structured Mesh (HASM) [38].

The uniform mesh refers to fixed patch size used in the mesh. Delaunay mesh is generated by using Voronoi diagrams. Voronoi diagram is a geometric dual of Delaunay mesh and one could be derived from the other. As a summary, the Delaunay mesh can be generated as follow: Given a set of N points in a plane, Voronoi diagram divides the domain into a set of polygonal regions, the boundaries of which are the perpendicular bisectors of the lines joining the points. Each tile contains only one of the N points. If both of the conditions are satisfied, the lines joining the points form the Delaunay mesh. The location of the node in Uniform mesh and Delaunay mesh are based on constrained algorithm, which could not capture the dynamics of the moving object, so they are not suitable for moving object segmentation.

Figure 2. 6. Uniform mesh
The Hierarchical Adaptive Structured Mesh (HASM) is comparatively a new technique. It is a coarse-to-fine adaptive mesh that captures the dynamics of moving object. The mesh is generated at its coarse level as a uniform mesh with a defined resolution. Then each patch with more motion components is divided into smaller patches. The division is implemented repeatedly until it reaches a maximum splitting level. The patches with the smallest size could represent the moving object. The HASM topology is shown in the figure below.

The Hierarchical Adaptive Merge/Split Structured Mesh (HAMSM) \([39][40]\), a refinement of HASM is a technique that constructs mesh topology using both coarse-to-fine and fine-to-coarse techniques. The mesh topology is generated through a split and
merge procedure. The process begins with the initial mesh structure; the patches with high prediction error are divided into four smaller patches evenly while every four patches with similar motion parameters are merged together to form a larger patch. The splitting and merging procedure are processed recursively until the criterion is satisfied or the maximum level is reached. Figure 2. 9 shows the result. The black nodes are newly generated nodes, and the dotted line connected nodes are eliminated nodes.

![Figure 2.9: Mesh topology after merge and split](image)

The HAMSM mesh is proved to be efficient both in compression and tracking field. It could largely reduce the load of mesh coding since the regions with little motion are merged. It could also be used in moving object tracking since it captures the dynamics of objects and ignores the subtle motions.

### 2.3 Video Object Tracking

Video object tracking is to find the trajectory or describe the dynamics of an identified video object with a sequence of images.

#### 2.3.1 Principle of object tracking
Let \( I(x_1, x_2, t) \) and \( I(x_1, x_2, t_0) \) denote the brightness values of the location \((x_1, x_2)\) in two frames acquired at time instant \(t\) and \(t_0\). Let the region of interest (object) to be tracked and defined as \( R = \{ r_1, r_2, ..., r_n \} \), where \( r_1, r_2, ..., r_n \) are sub-regions of an object. Then over time, the region of interest may shift and deform between frames of the video sequence. The object tracking problem is to identify and extract the motion of the region of interest as a function of time, which when applied to the resultant consecutive frames continuously will yield the object movements with certain assumptions, then the problem is simplified to estimating the parameters of the motion model.

### 2.3.2 Translational motion based tracking

A direct approach is to use simple linear block matching algorithms or hierarchical estimation algorithms to estimate the motion vectors for the regions segmented. A simple translation model is given by

\[
x' = x + M
\]

(2.21)

where \( x \) is the object on the previous frame, \( x' \) is the object on the current frame. \( M = (u, v) \) is a linear motion vector.

### 2.3.3 Affine model based motion tracking

An affine transformation [41][42][43] is an important class of linear 2-D geometric transformations. It maps variables (e.g. pixel intensity values located at position in an input image) into new variables (e.g. in an output image) by applying a linear
combination of translation, rotation, scaling and/or shearing (i.e., non-uniform scaling in some directions) operations.

Linear motion estimation based tracking may not yield efficient results in cases where non-rigid objects are involved and the regions of objects undergo motion like rotation. In such cases, affine motion models can be employed. Affine transformation is also called Affinity. This is selected to be part of our proposed real-time tracking algorithm, it will be detailedly described in chapter 3.
The zoom-invariant video object-tracking algorithm (ZIVOTA) is a motion-based object detection and tracking algorithm. ZIVOTA extracts object feature points and constructs an affine-based model to predict the size and position of an object during the tracking process. Affine models are zoom invariant, which makes it possible to track object with changing size and shape. Compared to traditional frame difference and optical flow methods, ZIVOTA largely reduces the computational cost because it explores only relevant object feature points and it processes smaller number of points instead of the full frame. Moreover, it is more accurate since affine transformation is viewpoint invariant.

ZIVOTA consists of two main stages. The first stage detects the object feature points and locates a set of object feature points that represents the video object. The extracted object feature points will be tracked in the consequent video frames. Motion-based object segmentation methods are used together with feature detector to avoid the influence of the background noise. Objects are detected using temporal difference and an object boundary box (OBB) can be located. The coordinates of the OBB are recorded and used for OBB reconstruction in the consequent frames to predict the changing size and position of the moving object. In the second stage, affine basis points are selected from
the detected object feature points. An affine model is built, and used to predict the OBB points in order to get the size and location of the video object.

### 3.1 Moving Object Detection

The proposed ZIVOTA uses region difference method, a reformed version of temporal difference [10]. An image is divided into regions, each region on the current frame will be compared with itself on the reference frame. If the difference is higher than a predefined threshold, this region is considered as changed, it could be a part of a moving object. It is described in the function below.

\[
    r_{i+1} = \begin{cases} 
        \text{foreground}, & \text{if } |r_i - r_{i+1}| > T \\
        \text{background}, & \text{otherwise}
    \end{cases}
\]

(3.1)

where \( r_i \) is a region on the reference frame, and \( r_{i+1} \) is the same region on the current frame. After all the regions are compared between two frames, connected regions are combined using the rule of eight-connectivity to form moving objects, it’s shown in Figure 3.1. Then the size of each identified object is estimated to filter the object from noise. The region-based method has the advantage that we do not have to deal with the single pixels. The temporal difference will result in pixels marked with “background” or “foreground” labels. In order to connect foreground pixels to form the moving objects, binary-connected-component Labeling should be used. Binary-connected-component labeling works on binary images. It scans an image, pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel regions, i.e. regions of adjacent
pixels which share the same set of intensity values. All pixels in the frame will be processed while much less regions will be processed in region difference method. It is obvious that region difference is much more efficient. Eight-connectivity neighborhood method [47] is used to combine the changed regions into individual semantically meaningful objects.

![Eight-connectivity neighborhood method](image)

Figure 3.1. Eight-connectivity neighborhood method

This technique can be used not only for connected regions labeling, but also for noise filtering. In case of significant noises, there will be disconnected regions all over the image. The result of eight-connectivity neighborhood method will be many isolated regions including both real moving objects and noises. By assigning a threshold of size, the small regions representing noises could be filtered out:

If \( \text{Size}[\text{object}(i)] > T \)

A video object (keep it)

Else

Noise (discard it) \hspace{1cm} (3.2)

where \( \text{size}[\text{object}(i)] \) calculates the size of \( i^{th} \) object, and \( T \) is the threshold, the value of which depends on the average object size that should be known in advance.
3.2 Feature Point Extraction

Feature points [31] are selected from the moving object and used for tracking in the video sequence. Corner detection methods [27][28][29] can provide good result of feature points on an image. If there is only one object in the image and the background is clear, the corner detector can produce reliable output. But if the background is textured and there are more than one objects exist, it is hard to decide which detected point belongs to an object and which one belongs to the background. As we described in the previous part, we use motion cues together with corner detector to give satisfactory results.

There are several kinds of different corner detection methods in the literature. We select Susan corner detector because it’s efficient and easy to use.

3.2.1 Susan Corner Detector

The basic concept of SUSAN (Smallest Univalue Segment Assimilating Nucleus) [31] corner detection is that each image point has associated with it a local area of similar brightness. A circular mask is placed on each image pixel, the brightness of each pixel within the mask is compared with the brightness of this mask’s nucleus, then an area of the mask can be defined which has the same brightness as the nucleus. This area conveys the most important information about the structure of the image in the circular mask. For a pixel to be a corner, the univalue area in the mask must be less than half of its maximum possible value.
Figure 3.2. Example of masks used in SUSAN feature detection

Figure 3.3. SUSAN masks with similarity colouring

Figure 3.2 shows a dark rectangle on a white background. A circular mask that has a center pixel which is known as the "nucleus" is placed at five image positions. After
comparing the brightness of each pixel within the mask with the brightness of this mask’s
nucleus, the USAN (Univalue Segment Assimilating Nucleus) result is shown in Figure
3.3, and SUSAN corner detector is described below in detail.

Digital approximations to circles have been used. The usual radius is 3.4 pixels,
which gives a mask of 37 pixels. It is shown in Figure 3.4.

![Digital approximation of SUSAN mask](image)

Figure 3.4 Digital approximation of SUSAN mask

The comparison of the mask center with each pixel within the mask is determined
by:

\[
c(\vec{r}, \vec{r}_0) = \begin{cases} 
1 & \text{if } |I(\vec{r}) - I(\vec{r}_0)| \leq t \\
0 & \text{if } |I(\vec{r}) - I(\vec{r}_0)| > t
\end{cases} \tag{3.3}
\]

where \( \vec{r}_0 \) is the position of the nucleus in the two dimensional image, \( \vec{r} \) is the position
of any other point within the mask, \( I(\vec{r}) \) is the brightness of a pixel, \( t \) is the brightness
difference threshold and \( c \) is output of the comparison. After the comparison is done on
each pixel within the mask, a running total, \( n \) is made:
\[ n(r_0) = \sum_r c(r, r_0) \quad (3.4) \]

This total \( n \) is the number of pixels in the USAN, it gives the USAN area. The parameter \( t \) controls the sensitivity of the feature detection. A value of 25 is suitable for most real images.

Next, \( n \) is compared with a fixed threshold, and \( g \) is the geometric threshold. For a "corner" to be present, \( n \) must be less than half of its maximum possible value, and it is equivalent to saying that if the nucleus lies on a corner then the USAN area will be less than half of the mask area, and will be a local minimum. The initial corner response is then created by using the following rule:

\[ R(r_0) = \begin{cases} g - n(r_0) & \text{if } n(r_0) < g \\ 0 & \text{otherwise} \end{cases} \quad (3.5) \]

where \( R(r_0) \) is the initial corner response. This is clearly a simple formulation of the SUSAN principle. The smaller the USAN area, the larger the corner response.

In feature extraction algorithms, there will always be at least one threshold which needs to be set up. This is so that the response given by feature enhancement can be used to distinguish between features and non-features. This is a restatement of the fact that to reduce the amount of data in the image, there must be at least one place in the algorithm where a "decision" or non-linear mathematical process occurs. Setting the value of the threshold is usually critical if the algorithm is to behave correctly. The dependence of the correct threshold on the data and the capability of setting it without human intervention are two factors which have a large effect on the success of the algorithm. In other words, if a threshold affects the quality of output of an algorithm in such a way that it is not
possible to set it automatically to the optimum value appropriate to the input data, then the algorithm is eventually of limited use, although it may appear to give fine results on individual (hand tuned) examples.

There are two main types of thresholds, which can be loosely labeled "quality" and "quantity". Most thresholds used in algorithms will fall partly into both categories, but often more into one than the other. The thresholds "g" and "t" are good examples of the two categories. The geometric threshold g clearly affects the quality of the output. Although it affects the number of "corners" found, but much more importantly, it affects the shape of the corners detected. For example, if it were reduced, the allowed corners would be sharper. Thus this threshold can be fixed (to the value previously explained) and will need no further tuning. Therefore no weakness is introduced into the algorithm by the use of the geometric threshold. The brightness difference threshold is very different. It does not affect the quality of the output as such, but does affect the number of "corners" reported. Because it determines the allowed variation in brightness within the USAN, a reduction in this threshold picks up more subtle variations in the image and gives a correspondingly greater number of reported "corners". This threshold can therefore be used to control the quantity of the output without affecting the "quality". This can be seen as a negative or a positive point. On the one hand, it means that this threshold must be set according to the contrast, noise etc. in the image, and on the other, this threshold can be easily tuned to give the required density of "corners". In practice, a value of 25 is suitable for most real images, and if low contrast images need to be catered for, the threshold can be varied automatically, to give a final number of reported "corners" which is appropriate to the higher level uses of the "corners". When SUSAN
was tested on an extremely low contrast image this threshold was reduced to 7 to give a “normal” number of corners. Even at this low value, the distribution was still good (i.e. not over-clustered) and the “corners” found were still quite stable.

The final stage in the SUSAN two dimensional feature detector is to search the initial response over square 5 by 5 pixel regions for local maxima (above zero). Therefore, the corner response image is suppressed so that non-maxima are prevented from being reported as corner points. It results in a list of features being reported. In summary then, the algorithm performs the following steps at each image pixel:

1. Place a circular mask around the pixel in question (the nucleus).
2. Use Equation 3.3 to calculate the number of pixels within the circular mask that have similar brightness to the nucleus. (These pixels define the USAN.)
3. Use Equation 3.5 to subtract the USAN size from the geometric threshold to produce a corner strength image.
4. Use non-maximum suppression to find corners.

The example of SUSAN corner detector output is shown in Figure 3.5.
3.3 Affine Motion Modeling And Tracking

Tracking in our work is achieved using affine motion model [42], a method which takes advantage of the viewpoint invariance of single image features and the collective temporal coherence of a cloud of such features, without requiring features to exist through entire sequences. The method is fundamentally invariant to zoom, and thus independent of errors in zoom. Furthermore, affine motion model also provides tolerance to features appearing at and disappearing from the edge of the image as a wider or narrower view is taken.
Two important requirements in visual surveillance are the ability to track and the ability to deal with zoom. A variety of general image features suggest themselves as candidates to be exploited for the tracking and zooming process, and their associated methods can be grouped into the three broad categories of region-, contour- and point-based.

For tracking alone, region-based methods [48], typified by correlation, suffer from the problem that they are not view-point invariant, and that they offer little immunity to local occlusions. Contour methods [19][21] are considerably better with respect to view-point invariance and occlusion insensitivity, but at the cost of incorporating prior templates. The ideal methods then would appear to be point-based. An image corner feature is viewpoint invariant, simple to extract and requires no complex model.

When zoom is introduced, the situation becomes more difficult for all categories of methods. Correlation now suffers particular badly because the method is fundamentally not invariant to zoom. Contour tracking, where the template is constrained to deform affinely, is invariant to zoom, but there is the practical problem of what happens if the single contour falls off the image while zooming-in. Again, a point-based method would seem the ideal. Under active fixation, the single point would always be near the image center and hence could not fall off the image.

Recent work has demonstrated the active tracking of clusters of corner features using the method of affine transfer [7][8]. The method finesse the difficulty caused by the temporal instability of a single corner by replacing the requirement to track one corner through all image frames by the less demanding requirement to track any three points across three successive frames.
Because scaling is an affine transformation, the method is fundamentally invariant to zoom. Moreover because the method allows corner to disappear and appear, and because the gaze points is not tied to a physical feature, it appears to solve the other problem introduced by zooming.

ZIVOTA [49] adopts point-based method using affine transfer for tracking. This newly proposed algorithm is different from the other point-based tracking methods in that it tracks not only the gaze point, but also the size of the target. The feature points detection is performed only when the correlation of the affine basis set between previous and current frame is less than a threshold. This results in reduction of computational cost.

3.3.1 Affine Transformation

An affine transformation [41] is an important class of linear 2-D geometric transformations. It maps variables (e.g. pixel intensity values located at position in an input image) into new variables (e.g. in an output image) by applying a linear combination of translation, rotation, scaling and/or shearing (i.e., non-uniform scaling in some directions) operations.

![Figure 3. 6. Affine transformation example](image)

The general affine transformation is commonly written in homogeneous coordinates as shown below:
\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  a_1 & a_2 \\
  a_4 & a_5
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  a_3 \\
  a_6
\end{bmatrix}
\] (3.6)

where \( a_i, i = 1, \ldots, 6 \) designate the parameters of the affine model, the base coordinates \((x, y)\) are transformed to produce the derived coordinates \((x', y')\).

Affine transformation can detect rotation, scaling, shear, etc. The six unknowns can be obtained by mapping the vertices of a triangle.

\[
a_1 = \frac{(x_1' - x_2') \times (y_1' - y_3') - (x_1' - x_3') \times (y_1' - y_2')}{(x_1 - x_2) \times (y_1' - y_3') - (x_1 - x_3) \times (y_1' - y_2')} \tag{3.7}
\]

\[
a_2 = \frac{(x_1' - x_2') \times (x_1 - x_3) - (x_1' - x_3') \times (x_1 - x_2)}{(x_1 - x_3) \times (y_1 - y_2) - (x_1 - x_2) \times (y_1 - y_3)} \tag{3.8}
\]

\[
a_3 = \frac{(x_1' - x_2' - x_1') \times (x_3 y_1 - x_1 y_3) - (x_1' - x_3') \times (x_2 y_1 - x_1 y_2)}{(x_2 - x_1) \times (x_3 y_1 - x_1 y_3) - (x_3 - x_1) \times (x_2 y_1 - x_1 y_2)} \tag{3.9}
\]

\[
a_4 = \frac{(y_1' - y_2') \times (y_1 - y_3) - (y_1' - y_3') \times (y_1 - y_2)}{(x_1 - x_2) \times (y_1 - y_3) - (x_1 - x_3) \times (y_1 - y_2)} \tag{3.10}
\]

\[
a_5 = \frac{(y_1' - y_2') \times (x_1 - x_3) - (y_1' - y_3') \times (x_1 - x_2)}{(x_1 - x_3) \times (y_1 - y_2) - (x_1 - x_2) \times (y_1 - y_3)} \tag{3.11}
\]
\[ a_6 = \frac{(y_1' x_2 - y_2' x_1) \times (x_3 y_1 - x_1 y_3) - (y_1' x_3 - y_3' x_1) \times (x_2 y_1 - x_1 y_2)}{(x_2 - x_1) \times (x_3 y_1 - x_1 y_3) - (x_3 - x_1) \times (x_2 y_1 - x_1 y_2)} \]  \quad (3.12)

where \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) are the coordinates of the vertices of a triangle on the reference frame. And \((x_1', y_1'), (x_2', y_2'), (x_3', y_3')\) are their coordinates on the current frame.

![Figure 3.7 A triangle distortion](image)

### 3.3.2 Zoom-Invariant Affine Motion Modeling and Tracking

While tracking an object undergoing a linear transformation, its position could be represented by the center of an object, which will be used as the gaze point to control the movement of the camera. Centroid is usually calculated to be the object center, and the contour of moving object is usually used to compute the size of the object. However, the calculation of centroid and contour is time-consuming. ZIVOTA uses object boundary box to represent the size of an object, and the center of object boundary box to replace the centroid.

As we described in chapter 3.1, the moving object is detected by region difference method, to speed up the operation, the region is selected to be a block. The changed
blocks will be connected to form objects according to the rule of eight-connectivity. Then the maximum object boundary box can be formed. The idea is shown below.

Figure 3. 8 object boundary box detection. The gray blocks are changed blocks and they form a moving object. The boundary box covers all the changed blocks.

The aim of our auto-tracking algorithm is to track the size and position of a moving object. The object size could be represented by the size of maximum object boundary box (OBB), and the object position could be located by the center of OBB, which we name it as gaze point. The object boundary box itself could be represented by its four corners: object boundary points. So only four points are needed to represent the object size.

Figure 3. 9 object boundary points and gaze point example. The cross is the object boundary center and also the gaze point, the 4 circles are the object boundary points.
When the object is detected, the boundary box can be located, then corner detection is performed within the boundary box. Three points are selected from the corner points, they will act as the affine basis points, \( \text{aff}_1 \), \( \text{aff}_2 \), and \( \text{aff}_3 \). Meanwhile, the locations of gaze point \( \mathbf{g} \) and the 4 corner points of the boundary box \( \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \) are recorded, and they will be reconstructed in the following frames. If the three affine basis points are matched in the next frame as \( \text{aff}_1', \text{aff}_2' \) and \( \text{aff}_3' \), the affine parameters can be computed, and equation (3.6) gives the new position of the gaze point \( \mathbf{g}' \), and the new object boundary points \( \mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}' \). Then in the third frame, a short time later, the affine basis points are tracked again and the affine parameters are computed again to reconstruct new object boundary points and gaze point. Neither the gaze point nor the object boundary points need to correspond to any physical features, they are all virtual points. Thus with any three corner correspondences (on the detected moving object) in two frames, we can reconstruct the positions of the desired points given their image coordinates in the previous frame. The three affine basis points need not to be the same over time, they will be abandoned if their correlation between 2 frames are less than the predefined threshold, and a new affine basis set will be selected. By using this method, there are only 3 corner points that will be tracked in every frame, and only 5 virtual points will be reconstructed to decide the object position and size.

### 3.3.2.1 Affine Basis Set Selection

Corners appear to be ideal features to provide information of the targets. However, there are still some problems we will have to face. One of the problems is: which point is more relevant?
ZIVOTA only extracts corner points within the object boundary box, which already partly solved this problem, but there will still be a small portion in the object boundary box which contains the background. ZIVOTA currently uses a technique based on corner velocity: it is assumed that the background is stationary and the target is moving, and there is only one moving object in the field of view. Every candidate corner point will be estimated for its velocity between frame $t_1$ and frame $t_2$. If its velocity is zero, it will be filtered out as a background corner.

Figure 3.10 shows that some background corners are detected together with the object's feature points. Taking into account the corner velocity, this problem could be solved.

Figure 3.10 Future points in object boundary box. The first picture gives the object boundary box, the second picture shows the detected corners within the object boundary box.

The second and considerably more important problem is that while tracking an individual corner, it may be only stable in a few frames. Either noise or occlusion will
inevitably cause it to disappear sooner or later. The possibility of fixating the tracking on any individual corner is ruled out. Individual corner disappearing and reappearing will significantly affect the affine tracking result.

A possible solution is to use a more sophisticated control strategy. We select individual corners to be affine basis point, and switch from one set to another set when they disappear. Figure 3.11 shows this idea. The black solid circles are the affine basis points, and other circles are feature points.

Figure 3.11 Selecting different affine basis set from feature points.
A rapid stable selection of affine basis set can be made by automatically choosing the left and right most feature points on the moving object, and then choosing the third point that maximizes the area of the triangle defined by the three points.

The third and the most difficult problem is in case of tracking a non-rigid object like a running person or a speed skater. For example, while tracking a skater, if we just randomly select 3 points from all the feature points detected on the moving skater to be the affine basis points, the selected points may be on one leg or arm, which could not represent the global motion of the skater and will get wrong prediction results. So we must set some constraints for the selection of the feature points.

Head and feet motions of the skater should be the most suitable to represent the global motion of a skater. The constraints should be set to locate the 3 selected affine basis points on head and feet. Locating the feet is not difficult. We can do that by limit the searching area to be around the lower left and right corner of the detected object boundary. But for the head, the situation is more complex. Since the skater is swinging his arms, the head is not always the top. We use the vertical projection of the binary difference image to locate the head. Because of the structure and shape of the skater, it is likely that the projection of the head will remain the extreme point even if it’s actually not the uppermost part of the body.

The projection-based affine basis set selection has 3 steps. First, the skater is detected using frame difference algorithm. Then morphological filtering is applied to eliminate noise and enhance the difference image. Finally, vertical projection histogram will be computed by projecting the binary difference region on a horizontal axis. In order to save computation, morphological filtering and projection are done within the detected object
boundary. In this way, the movement of skater’s arms will not be taken into account, and the tracking result is much more precise.

![Images](image1.jpg)

**Figure 3.12.** Example of head selection. (a). The original picture. (b). The frame difference result. (c). Morphological filtering output. (d). Vertical projection.

### 3.3.2.2 Motion Estimation Of Feature Points

The affine basis set will be tracked in every frame to generate different affine parameters. There are several kinds of motion estimation algorithms, such as Three Step...
Search (TSS), Full Search, Cross Search, and Optical Flow. Motion estimation algorithms rely on the fundamental idea that the luminance \( s \) of a point \( p(x_1, x_2) \) on a moving object remains constant along \( p \)'s motion trajectory, represented as

\[
s(x_1, x_2, t) = s(x_1 + d_1(x), x_2 + d_2(x), t + \Delta t) \quad (3.26)
\]

where \( x_1 \) and \( x_2 \) denote the motion of the point \( p \) within the time \( \Delta t \).

In ZIVOTA, the motion vectors (MV) of the affine basis points are estimated by generating an 8x8 block of pixels surrounding the point as shown in Figure 3.13. Then by applying the motion estimation method on the whole block to generate the motion vector.

![8x8 block diagram](image)

**Figure 3.13.** Motion estimation of affine basis points.

The displacement of the image-plane coordinates \( x \) from time \( t \) to \( t' \), is referred to as a correspondence vector \( d(x,t) \). An optical flow vector \( v(x,t) \) is defined as the temporal rate of change of the image-plane coordinates, \((v_1, v_2) = (dx_1/dt, dx_2/dt) \) at a particular point \( (x,t) \in \mathbb{R}^3 \) as determined by the spatio-temporal variations of the
intensity pattern $s_c(x,t)$. That is, it corresponds to the instantaneous pixel velocity vector.

The 2-D motion estimation problem can be posed as either:

1) the estimation of image-plane correspondence vectors
   
   $d(x,t;\Delta t) = [d_1(x,t;\Delta t) \ d_2(x,t;\Delta t)]^T$ between the times $t$ and $t + \Delta t$, for all $(x,t) \in \Lambda^3$, or

2) the estimation of the optical flow vectors $v(x,t) = [v_1(x,t) \ v_2(x,t)]^T$ for all $(x,t) \in \Lambda^3$. The correspondence and optical flow vectors usually vary from pixel to pixel, due to rotation of objects in the scene, and as a function of time, due to acceleration of objects.

The correspondence problem can be set up as a forward or backward motion estimation problem, depending on whether the motion vector is defined from time $t$ to $t + \Delta t$ or from $t$ to $t - \Delta t$, as depicted in Figure 3.14 and Figure 3.15.

Forward Estimation: Given the spatio-temporal samples $s_p(x,t)$ at times $t$ and $t + \Delta t$, which are related by

$$s_p(x_1,x_2,t) = s_p(x_1 + d_1(x,t;\Delta t),x_2 + d_2(x,t;\Delta t),t + \Delta t) \quad (3.13)$$

or, equivalently,

$$s_k(x_1,x_2) = s_{k+l}(x_1 + d_1(x),x_2 + d_2(x)) \quad (3.14)$$

find the real-valued correspondence vector $d(x) = [d_1(x) \ d_2(x)]^T$, where the temporal arguments of $d(x)$ are dropped.
Backward Estimation: If we define the correspondence vectors from time $t$ to $t - \Delta t$, then the 2-D motion model becomes

$$s_k(x_1, x_2) = s_{k-1}(x_1 + d_1(x), x_2 + d_2(x)) \quad (3.15)$$

Alternatively, the motion vector can be defined from time $t - \Delta t$ to $t$. Then we have

$$s_k(x_1, x_2) = s_{k-1}(x_1 - d_1(x), x_2 - d_2(x)) \quad (3.16)$$

ZIVOTA uses forward estimation and Three Step Search with SAD matching criterion.
3.3.2.3 Tracking The Object

After finding the affine basis set on the next frame, affine parameters are acquired, then the new location of the object (the gaze point) and the new size of the object (object boundary points) can be reconstructed, which is shown in Figure 3.16.

The motion vector (MV) of the object can be calculated:

\[
SAD(dx, dy) = \sum_{m=x}^{x+N-1} \sum_{n=y}^{y+N-1} |I_k(m,n) - I_{k-1}(m+dx, n+dy)|
\]  

(3.17)

Figure 3.16. Affine object tracking
\[ MV(x, y) = g_e(x_c, y_c) - g_r(x_r, y_r) \] (3.18)

\( g_e(x_c, y_c) \) is the coordinate of the gaze point on the current frame, and \( g_r(x_r, y_r) \) is the coordinate of the gaze point on the reference frame.

Unlike other tracking methods such as correlation matching, this method is viewpoint invariant, and will work even if the fixation point is occluded. As long as the affine basis are matched between frames, the gaze point and object boundary points can always be reconstructed.
Figure 3.17. The change of affine basis set

From Figure 3.17 we can see, when there is partial occlusion, the current affine basis points cannot be tracked on the next frame, so ZIVOTA will reselect a new affine basis set. The gaze point and object boundary points need not to be visible, they can be reconstructed even though the corresponding physical object points are occluded.
CHAPTER 4

PROTOTYPE AND EXPERIMENTAL RESULTS

4.1 Prototype System

ZIVOTA is prototyped using an active camera system (Canon VCC4), a camera switch, a video capture card and a personal computer which has a 1.8GHz Pentium processor. It is shown in Figure 4.1.

![Prototype System Diagram](image)

Figure 4.1 Prototype system

The graphic user interface of the prototype is shown in Figure 4.2. Live video preview is captured and shown on the left side, the tracking status and result are shown on the right side.
A camera coordinate system is used to represent image pixels. Each pixel in an image frame taken by VCC4 camera can be defined as \((x, y, \alpha, \beta, \delta)_{ij}\). \(x\) and \(y\) represent the image pixel location (the (0,0) point of the coordinate is the center of the whole image), \(\alpha\) is pan angle, \(\beta\) is tilt angle, and \(\delta\) is the zoom setting.

The camera is first set to a default setting, and begins to do foreground segmentation. If there is any moving object appears within the camera’s current field of view, it will be detected and tracked. The object gaze point is set to be the camera gaze point (the center of the image). During the presence period of this object, the gaze point is predicted from frame to frame by affine motion model (this will be discussed in detail later). The camera is controlled to follow the detected gaze point to keep the moving object in the center of
the camera’s field of view. Assume current location of the object gaze point is already
the camera gaze point, it could also be represented by CENTER (0, 0), and \((x_{f+1}^p, y_{f+1}^p)\) be
the predicted location of the gaze point on the next frame, the new camera setting could
be computed as follow:

\[
\begin{align*}
\text{Pan position} & \quad \alpha_{i+1} = \alpha_i + f(\delta) * (x_{i+1}^p - \text{CENTER}_x) \\
\text{Tilt position} & \quad \beta_{i+1} = \beta_i + f(\delta) * (y_{i+1}^p - \text{CENTER}_y) \\
\text{Zoom position} & \quad \delta_{i+1} = \delta_i
\end{align*}
\] (4.1)

\(\alpha_{i+1}, \beta_{i+1}, \delta_{i+1}\) represent the new camera location. \(f(\delta)\) is the pan/tilt angle factor under
zoom setting \(\delta\). The new object gaze point will be the center of the next image after the
camera moves to the new pan, tilt and zoom position. Since we first deal with the pan/tilt
control, the zoom setting is assumed to be constant in frame \(i\) and frame \(i+1\). This is
shown in Figure 4. 3.

![Figure 4. 3. Example of camera position control.](image-url)
The cross in Figure 4.3 represents the object gaze point, the circle represents the camera gaze point CENTER (0, 0). In the first frame, the object gaze point and the camera are overlapped. In the second frame, they are in different image locations. The purpose of pan/tilt control is to force them overlap again.

The zoom operation is performed when the detected moving object is too small or too large. Object will be zoomed in or out to the desired size. The object boundary is predicted by affine motion model, and then the size of the predicted object will be calculated and compared with the desired object size. If the difference ratio exceeded a predefined limitation, the new zoom setting will be computed, and the camera will be controlled to zoom into the new setting to hold the object in its field of view. Let $s_d$ be the desired object size after zooming, and $s_{i+1}$ be the predicted object size on the next frame, the new zoom setting is calculated as follow.

$$
\delta_{i+1} = \delta_i + g\left(\frac{s_d}{s_{i+1}}\right)
$$

(4.2)

$\delta_i$ is the previous zoom setting, and $\delta_{i+1}$ is the new zoom setting. $g\left(\frac{s_d}{s_{i+1}}\right)$ is the function used to transfer size ratio into zoom steps.

The camera switch has eight video input and one output. It is controlled by the computer's parallel port to switch between 8 cameras, so the target can be tracked from different point of view.

The video image processing functional blocks are shown in Figure 4.4. Images will be captured from the output of camera system, the moving object will be modeled and tracked, the tracking result will be fed back to control the movement of camera to keep
the moving object in the camera’s field of view. At the same time, captured images will be previewed on the display.

Figure 4. 4. Video signal processing block diagram

4.2 Off-line Experimental Results

In off-line experiments, the active camera system is not used. The purpose of this experiment is to test the functionality and accuracy of ZIVOTA.

4.2.1 Results of “robot” sequence

A robot is moving towards the camera. The original sequence is shown in Figure 4. 5.
Figure 4. 5. The “robot” sequence showing a toy car moving towards the camera

The moving object is detected using region difference method, which is the first step of ZIVOTA. The object boundary box can then be located.

Feature points detection is conducted within the object boundary box. The result is shown in Figure 4. 7. Solid white boxes are the detected feature points, they are enlarged in order to be shown clearly.
Figure 4. 8 shows the affine basis set selected from feature points. The three white solid blocks are the selected affine basis point. White points are the feature points.

(a) region difference  
(b) object boundary box

Figure 4. 6. Moving object detection on “car” sequence

Figure 4. 7. Feature points detection result
Figure 4.8. Affine basis points selected from feature points

(a) frame 1
(b) frame 11
(c) frame 21
(d) frame 31

Figure 4.9. ZIVOTA tracking result of “car” sequence
The final tracking result of ZIVOTA is shown in Figure 4.9. The cross is the reconstructed gaze point and the box represent the object boundary box. Since the reconstructed object boundary points may not be able to form an exact rectangle, the object boundary box is decided by minimizing a rectangle to contain 4 object boundary points.

4.2.2 “speed skater” sequence

Tracking of a speed skater is different from tracking of a car in that the speed skater is a non-rigid moving object. Projection-based affine basis set selection is used.

Figure 4.10. The skater sequence
The first step is also moving object detection using region difference. Noise regions are filtered out.

(a) region difference

(b) object boundary box

Figure 4.11. Moving object detection on “skater” sequence
Figure 4. 12. Frame difference

Figure 4. 13. Head position detection using projection. (a) Morphological filtering output of foreground region, (b) detected moving object, (c) vertical projection histogram

Figure 4. 14. Affine basis selection on "speed skater" sequence
One of the affine basis points is selected near the vertices of vertical projection histogram, the other two affine basis points are located on the feet by limiting the searching area to be around the lower left and right corner of the detected object boundary.

Figure 4.15. ZIVOTA tracking result of “speed skater” sequence
Figure 4. 15 shows that ZIVOTA successfully tracked the moving speed skater. By using projection-based affine basis selection method, the selected affine basis points are always on head and feet. It makes the tracking result stable.

4.3 Real-time Experimental Results

The real-time experiments have been conducted using the prototype system.

(f) frame 30

Figure 4. 16 is the sequence captured while doing real-time tracking. The gaze point (the white cross) is always near the frame center, which is the desired result.
Figure 4. 16. Real-time experiment result captured through the lens of active camera

Figure 4. 17 shows the experimental results while a robot is moving towards the camera. In the first frame, the car is detected and judged to be too small, ZIVOTA controls the active camera to zoom in to get sufficient resolution. When the object appears too big, ZIVOTA controls the camera to zoom out. The camera is also controlled to pan and tilt to follow the robot.
The tracking result when there is partial occlusion is also shown. From Figure 4. 18 we can see, the moving object is still tracked and its position is kept near the frame center also it’s partially occluded.
4.4 Performance Analysis

The performance of ZIVOTA is evaluated using off-line video sequence and under real-time situation. ZIVOTA runs at 15 to 65 Hz on a PC which has a 1.8GHz CPU. When there is camera movement, additional time of averaging 300ms will be added to communicate between the camera and PC, and wait for the camera movement to stop.
4.4.1 Computational Cost

In real-time applications, processing time is crucial, so the evaluation of computational cost is very important. ZIVOTA computational cost consists of three parts: moving object detection cost, feature points detection cost and affine prediction cost.

Compared to the temporal difference foreground detection techniques, the region difference method is advantageous because we don’t have to deal with the single pixel. The pure temporal difference will result in pixels that is marked with “background” or “foreground” labels. In order to connect foreground pixels to form the moving objects, binary-connected-component will be used, all pixels in the frame need to be processed. If the input image is 320 x 240 pixels, 76800 pixels will be processed while only 1200 blocks will need to be processed if the block is 8 x 8 pixels. It is obvious that block difference is much more efficient.

Feature points extraction is the most time-consuming stage, since a pixel has to be compared with every pixels within a mask. By only dealing with pixels inside the object boundary box, the computational cost of this stage will be largely reduced. For example, if the object boundary box size is 10 x 10 blocks, it consists of 6400 pixels. Compared with the whole frame which has 76800 pixels, there is less than 1/10 of the original computation.

Other point-based tracking algorithms found in the literature like Reid and Murray’s work normally perform feature points detection on every frame, and do correlation matching between each frame. It is too time consuming to be embedded into any real-time tracking system without additional hardware components, and actually they used more than one microprocessors in their implementation. ZIVOTA selects three feature
points and only tracks them between frames. Feature points detection will not be done again if the matching correlation value is above a predefined threshold. The analysis of processing time is shown in Figure 4. 19 and Figure 4. 20. Figure 4. 19 shows the ZIVOTA tracking result of a car moving from left to right, where Figure 4. 20 shows the timing analysis.

Figure 4. 19. ZIVOTA tracking results of a robot moving from right to left

(sequence1)
Since moving object detection and feature points extraction will only be done when the affine basis set can’t be matched on the next frame, they will happen periodically. The peaks in Figure 4. 20 indicate the processing time of moving object detection and feature points extraction, it’s about 61 milliseconds. While the affine basis set is matched between frames, about 18 milliseconds per frame will be used. The average processing time is about 22 ms/frame.

4.4.2 Measuring Confidence

Measuring confidence is a quantitatively evaluation of performance. It is computed by examining the likelihood values of the tracking results. ZIVOTA tracking result has two states, the object position and size, and each state has an associated likelihood value. To compute measuring confidence, the two likelihood curves are averaged.
To calculate the likelihood value, the object size and position should be acquired. Figure 4.21 shows the example of predicted object size and trajectories.

(a) the robot moves along an ellipse track (sequence 2)
Figure 4. Demo of predicted object trajectory and size

(b) the predicted trajectory of the robot

(c) the predicted size of the robot

The detailed analysis will be given based on sequence 1. Since the robot moves straight from left to right, it's easy to identify the real size and trajectory of the robot.
Figure 4. 22. ZIVOTA predicted trajectory of the gaze point (the video sequence used is the same as Figure 4. 19, sequence I)

Figure 4. 22 shows the object trajectory of ZIVOTA tracking. The predicted moving object trajectory is almost overlapped with the real trajectory, which is carefully identified by eye. The maximum error is 7 pixels, which is acceptable.
Figure 4. 23 shows the object boundary box and its size predicted by ZIVOTA. The real size of the object boundary box is identified manually.

Figure 4. 23. ZIVOTA predicted size of the moving object (sequence I)

The likelihood curve, which we name it as measuring confidence, is computed by the following equations:

$$MC = \frac{V_{\text{real}} - Err}{V_{\text{real}}}$$  \hspace{1cm} (4.1)
where MC is the measuring confidence, $V_{\text{real}}$ is the real size/position, $Err$ is the error between real and predicted size/position value.

$$Err = |V_{\text{real}} - V_{\text{predicted}}|$$  \hspace{1cm} (4.2)

Figure 4.24, Figure 4.25 and Figure 4.26 are the position confidence, size confidence, and measuring confidence of sequence I.

![Figure 4.24](image1.png)

**Figure 4.24.** The size prediction confidence of ZIVOTA (sequence I)

![Figure 4.25](image2.png)

**Figure 4.25.** The position prediction confidence of ZIVOTA (sequence I)
The size confidence values are lower than position confidence values because size are decided by the four object boundary points, so the accumulated errors are four times more than the position error. The measuring confidence values (between 87% to 99%) indicate the object in sequence I is successfully tracked.
CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Summary

A proposed real-time zoom-invariant video object tracking algorithm (ZIVOTA) has been described. It can be used to track human or rigid-shaped object. ZIVOTA can provide information of object-of-interest while performing tracking, which enables automatic visual system control. It employs the same set of features to track both the position and size of a moving object through frames. Time-consuming processes like feature points detection and affine basis set selection will be done only when it's necessary. Computational load is largely reduced. By using affine motion model, this technique is fundamentally zoom-invariant, it can deal with translation, rotation, and scaling of objects. Since the gaze point (represents object location) and object boundary points (represent object size) are virtual, ZIVOTA is able to handle partial occlusion.

ZIVOTA is successfully implemented by C++ and runs on a prototype system consisted of an active video camera and a personal computer. Our experiments have demonstrated the ability of ZIVOTA to track non-rigid moving object and control an active camera to zoom, pan and tilt to keep the object in the captured video frame with sufficient resolution.

ZIVOTA has the potential to be used in several kinds of applications, such as coaching tools, security surveillance, videoconference and video-based education.
5.2 Discussion And Future Work

Although the real-time experimental results show ZIVOTA has very good performance of tracking, there are still some issues affecting the stability of the process.

The pan/tilt speed of the Canon VCC4 camera that we are using is about 13ms/degree. They are acceptable compared with the working speed of ZIVOTA (30 to 37 Hz). But the time required for communication between the active camera and the computer is about 230ms, which will terribly affect the performance of the whole system while any camera movement is required. Velocity parameter is added into the object motion model to solve this problem and the system performance is improved. Acceleration parameter will be considered in the future work. Periodical motion analysis may also be a potential to increase the tracking accuracy.

ZIVOTA currently has different versions of application to track rigid-shaped object like a car or non-rigid shaped object like a human. If we use some identification techniques to classify targets into different categories, ZIVOTA can automatically shift between its versions.

Multi-object tracking is under consideration. Several appearance models of different moving objects can be set up and updated in the process of tracking.
REFERENCES


International Workshop On Image Processing, real-time edge and motion


Detection”, Proc. European Signal Processing Conference, pp. 2549-2552, Island of
Rhodes, Greece, 1998.


Technical Report,University of Western Australia,Department of Computer

[33] A. Singh and M. Shneier, “Grey level corner detection: A generalization and a robust
real time implementation”, Computer Vision, Graphics and Image Processing
51:54-69, 1990.


