Computer Vision Applications for Nanotechnology Research and Surveillance System Design

by

Steven Cheng

(Under the direction of Suchendra M. Bhandarkar)

Abstract

The field of computer vision comprises various disciplines including topics such as background modeling, object tracking, and object detection. The research contained in this manuscript investigates two important aspects of computer vision, namely contour tracking and static object detection. These areas are elements of the more generic process of object analysis. The material discussed within, originated from the requirements of two unique research groups at the University of Georgia.

The Nanoscience group in the Physics and Astronomy department required a tracking application for their research on dynamic wettability of nanostructured surfaces. The initial specifications required a software program that could analyze recorded videos of a water droplet spreading on di erent surfaces fabricated from nanotube arrays. The *Contour Tracker* utilizes advanced object tracking techniques to track the outer contour of the water droplet. *Part* I outlines the contour tracking problem and the proposed *Contour Tracker* solution.

The Visual and Parallel Computing Lab has many projects in their repertoire including research on surveillance and monitoring system design. A common goal of surveillance systems is to detect stationary objects within the scene. A static object could indicate several

scenarios depending on the location of the scene. For instance, an idle object in an airport may be someone's forgotten luggage or worse, an explosive device, whereas static objects on the side of the street could indicate an illegally parked vehicle. *Part* II describes a simple and flexible method for static object detection.

INDEX WORDS: Active Contours, Snakes, Kalman Snake, Contour Tracking, Multiscale Background Model, Static Object Detection, Surveillance System

Computer Vision Applications for Nanotechnology Research and Surveillance System Design

by

Steven Cheng

B.Eng., McGill University, 2002

Montréal, Québec, Canada

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2006

Computer Vision Applications for Nanotechnology Research and Surveillance System Design

by

Steven Cheng

Approved:

Major Professor: Suchendra M. Bhandarkar

Committee: Walter D. Potter
Khaled Rasheed

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2006

Dedication

This thesis is dedicated to my family for their eternal love and support and to my friends for making this a wonderful and memorable journey.

Acknowledgments

"Exponential growth looks like nothing is happening, and then suddenly you get this explosion at the end,"

— Ray Kurzweil

No truer words could be said to describe my time at the Artificial Intelligence Center at The University of Georgia. I arrived in Athens with no notion of what AI-ology was all about. But for the past 3 years, the AI Center faculty has instilled in me a great deal of knowledge and I feel that I have grown immensely from this experience. I hope that someday I will be as influential to other people as they were to me.

Many thanks to everyone who has played a role in this portion of my life, especially: Dr. Bhandarkar for being my major professor and his continuous patience and understanding during my thesis adventure. Dr. Potter for imparting his knowledge and for teaching me ways to think outside of the box. And for his cool and rad way of teaching. Dr. Rasheed for his participation on my committee.

I would also like to show my appreciation to my fellow AI students for the long nights of programming and interesting discussions: Hendrik Fischer, thank you for your constant encouragement. Darren Casella, thank you for the fascinating stories of your life experiences and for teaching me how to ride a motorcycle. Vineet Khosla, thank you for being my HCI programming partner. Xingzhi Luo and Jianguo Fan, thank you for helping me finish my thesis. And thank you to: Sarah Witzig, Makiko Kaijima, Srigopika Radhakrishnan for your friendship. And thank you to Christina Kim for proof-reading this entire manuscript.

And last, but not least, thank you to my amazing family for their support (financially and otherwise) throughout my career at The University of Georgia.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Dynamic wettability of a nanostructured surface is an important property for many liquid related applications of nanostructures. In *Part* I of this manuscript, a software application employing an active contour model is developed to analyze the evolution of the precursor (outer rim) boundary of a water droplet as it spreads on a nanostructured surface. Experiments show reasonable agreement between the results of the *Contour Tracker* and those obtained via manual measurements.

In *Part* II, a multiscale background scheme is applied to dynamic scene analysis. The length of time that an object remains stationary in a scene is a common feature provided by computer monitoring systems. For instance, in an airport environment, an object observed to be unattended for an abnormal amount of time could indicate a security threat or someone's left-luggage. The extraordinary size of such buildings makes detection a tedious and error-prone activity for humans. By automating the detection process, the overall monitoring system benefits from greater coverage and quicker response times. The results show that a multiscale approach to static object detection performs well in indoor situations.

Part I

Image-based Metrology of a Water Droplet Spreading on Nanostructured Surfaces [1]

[1] S. Cheng, J. G. Fan, X. Luo, S. M. Bhandarkar, and Y. P. Zhao. 2006. Submitted to 2006 IEEE Transactions on Nanobioscience.

Chapter 2

Introduction

Wettability measurement of a solid surface is a topic of great importance in a variety of problem areas ranging from ink-jet printing, painting, corrosion, biofoiling, DNA immobilization, cell growth, and tissue engineering. The wettability of a surface is a ected by both the surface chemistry and topography. It has been demonstrated that a micro/nano patterned surface can greatly alter the surface wettability such that a hydrophobic surface becomes more hydrophobic and a hydrophilic surface becomes more hydrophilic [13]. Moreover, superhydrophobic surfaces (contact angle $> 150^{\circ}$) have been achieved by treating nanostructured surfaces with chemicals which have low surface energies [10]. Lau et al. have found that a water droplet could easily spread on an as-grown carbon nanotube array. But once the array was coated with polytetrafluoroethylene (PTFE), the surface exhibited superhydrophobic behavior. The contact angle of the PTFE coated carbon nanotube array was observed to increase as the height of the carbon nanotube array increased. The surface contact angle was observed to provide a quantitative measure of the *static* wettability properties of the surface.

In addition to the measurement of static surface wettability properties, namely the contact angle described above, the dynamic spreading of a water droplet on flat, rough or porous surfaces has also been studied quite extensively [1, 2, 3, 4, 7, 17, 19]. Most investigations have addressed the evolution of a liquid droplet over time, and it is believed that the radius of the droplet R versus time obeys a simple power law, $R \propto t$, where the characteristic of a spreading regime due to a specific underlying physical mechanism. For example, for a flat surface, usually takes a value between $^1/_{10}$ and $^1/_8$, depending on whether the dominant driving force is the capillary force or the gravitational force [1, 2, 3, 4, 6, 7, 17, 19].

For a rough or porous surface, the spreading process becomes more complex and therefore, parameters such as the roughness or porosity of the substrate and the viscosity of the liquid have to be considered.

So far, there is still no systematic study of the dynamic spreading of liquids on nanostructured surfaces [5]. Depending on the detailed surface structure, one could treat a nanostructured surface either as a rough surface or as a porous surface. It would be interesting to know if the spreading of a liquid droplet on such nanostructured surfaces follows a similar dynamic scaling law as is observed for rough or porous surfaces. The study has the potential to provide useful surface wettability information for applications that involve the interaction of liquid droplets with nanostructures. For most dynamic spreading study, CCD cameras are used to track the evolution of the water droplet on the surface. This kind of image/video-based metrology o ers a non-contact and nondestructive means to study the wettability properties of nanosurfaces. One of the primary advantages of image/video-based metrological techniques is that they entail minimal or no perturbation of the physical quantity being measured. However, one of the major challenges posed by image/video-based metrological techniques in the study of the dynamics of the spreading of a water droplet on a nanostructured surface is the prohibitively large data processing requirement. In our study, the spreading of water droplets on Silicon nanorod arrays is recorded using a CCD camera with a very high shutter speed, capable of acquiring 210 frames per second (fps) at VGA frame resolution (800×600 pixels). For a video recording of $\sim$10 seconds, which is the typical time frame for a water droplet to spread completely on a wettable surface, one obtains over 2000 video frames of data. Although the study of the scaling law $R \propto t$ may not require one to necessarily analyze all the video frames, it would be advantageous to analyze as many frames as possible thereby taking full advantage of the fast CCD camera. Moreover, for obtaining further information such as the instantaneous droplet spreading speed $v = \Delta R/\Delta t$, the smaller the time interval Δt, the more accurate the speed computation. Accurate measurement of the instantaneous droplet spreading speed as a function of time would therefore

entail the processing and analysis of a large number of video frames. Manual processing and analysis of such large amounts of video data is onerous, error-prone and simply prohibitive. Thus, computer vision techniques for the automated processing and analysis of the video data are called for. In this paper we describe the design and implementation of computer vision algorithms which can automatically extract and track the shape and size (dimensions) of the spreading droplets in the video frames.

Chapter 3

Tracking of Droplet Contours in Video Frames

Figure 3.1 shows a typical video frame depicting the spreading of a water droplet on a nanosurface comprising of Silicon nanorods. The inner contour of the droplet, termed the *contact line*, denotes the position of the water droplet above the Silicon nanorods. The outer contour termed the *precursor*, denotes the frontal envelope of the water droplet as it spreads inside the nanorod channel. Although the outer contour of water droplet in Figure 3.1 is almost circular, it could potentially deform to any shape depending the hydrophobic/hydrophilic properties and the isotropic/anisotropic nature of the nanosurface. Consequently, we model the outer contour of the water droplet using a snake which is an active contour model capable of modeling a closed contour of irregular geometry. A snake

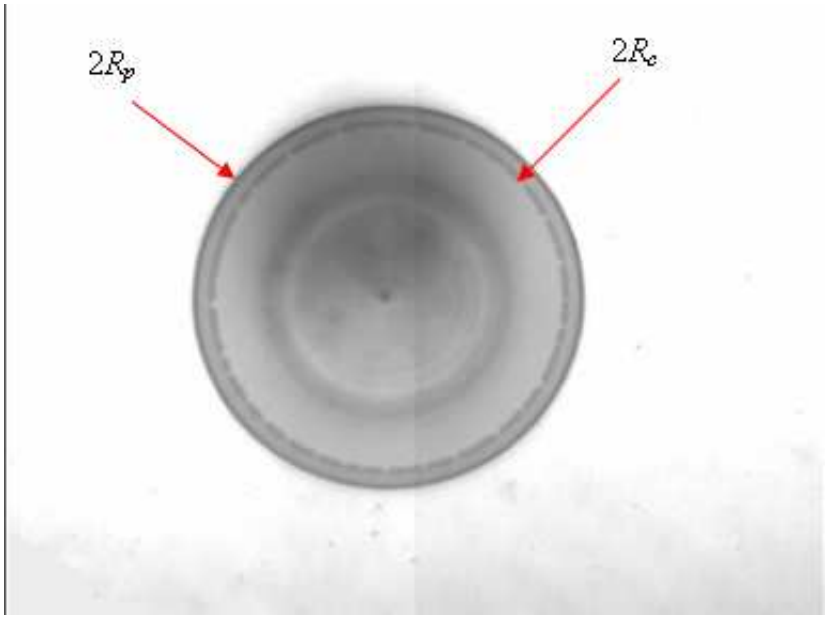

Figure 3.1: A single frame of the video showing a water droplet spreading on a Silicon nanorod surface. R_c and R_p are the radii of the contact line and precursor, respectively.

is an energy-minimizing spline proposed by [9] for the purpose of modeling contours of irregular geometry. Snakes are guided by external constraint forces and influenced by image forces that pull it towards features such as lines and edges. Snakes are active contour models in the sense that they lock onto features in their spatial vicinity and continuously track the locations of the features via a process of energy minimization. A snake is represented by a set of points or snake elements (snaxels) where the local and global position of each snaxel is determined by an energy function. The energy function contains terms that represent both, the external constraint forces and the internal image forces. The goal is to use the snake to find the desired image curve or contour by minimizing the energy function.

3.1 Snake Energy Function

The snake energy function is used to localize each snaxel within the image. Typically, the energy function is divided into two primary components: external energy and internal energy. The external energy is governed by image forces that attract the snaxels towards high-level image features such as points, lines, and edges [9]. Since our research focuses primarily on image contours, the natural choice for the external energy term is based on an edge function computed using an edge detector. The internal energy, on the other hand, determines the overall contour shape. The internal energy manipulates the snaxels location relative to adjacent snaxels (local positioning) and generally consists of first-order and second-order terms. The first-order term influences the snakes continuity while the second-order term influences the curvature of the snake.

The total energy $E(S)$ of the snake S is defined as the sum of the total energy value $E(p_i)$ (internal energy + external energy) of each snaxel p_i as in Equation 3.1.

$$\begin{aligned} E(S) &= \sum_{i=1}^{N} E(p_i) \\ &= \sum_{i=1}^{N} E_{Internal}(p_i) + E_{Extenal}(p_i) \\ &= \sum_{i=1}^{N} \quad (p_i)E_{Continuity}(p_i) + \quad (p_i)E_{Curvature}(p_i) + \quad (p_i)E_{External}(p_i) \end{aligned} \tag{3.1}$$

where the weights $\quad(p_i)$, $\quad(p_i)$, and $\quad(p_i)$ are vectors of real-valued numbers representing the contribution of each energy term to the overall energy of snaxel p_i. A simplified version of the energy function uses a snaxel-independent (constant) set of weights (i.e. for all $i \neq j$, $\quad(p_i) = \quad(p_j) = \quad$, $\quad(p_i) = \quad(p_j) = \quad$, and $\quad(p_i) = \quad(p_j) = \quad$). In summary, the energy function is used to locate (external energy) and shape (internal energy) the snake within an image.

The continuity energy focuses on spreading the snaxels evenly along the snake and is defined in Equation 3.2.

$$E_{cont} = (\bar{d} - \|p_i - p_{i-1}\|)^2 \tag{3.2}$$

where $\bar{d}$ is the average distance between snaxels and $\|p_i - p_{i-1}\|$ is the Euclidean distance between snaxels p_i and p_{i-1}.

The curvature energy attempts to minimize the number of oscillations along the snake and is given by Equation 3.3.

$$E_{curv} = \|p_{i-1} - 2p_i + p_{i+1}\|^2 \tag{3.3}$$

The external energy E_{ext} attempts to position the snaxels in the proximity of edge pixels in the image $I(x, y)$ where the magnitude of the image intensity (grayscale) gradient $\nabla I(x, y)$ is high and is given by Equation 3.4.

$$E_{ext} = -|\nabla I(x, y)|^2 \tag{3.4}$$

From Equation 3.4 it can be seen that high grayscale gradient magnitude values result in lower values for the external energy E_{ext}. The discrete grayscale gradient is computed using a pair of convolution kernels to approximate the image intensity derivatives in the vertical and horizontal directions. Equation 3.5 shows the vertical and horizontal kernels of a basic gradient operator.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \; G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{3.5}$$

Let $I_x(x,y) = I(x,y) \bullet G_x$ and $I_y(x,y) = I(x,y) \bullet G_y$ where $\bullet$ denotes the convolution operator. The image intensity gradient magnitude $|\nabla I(x,y)|$ is given by Equation 3.6.

$$|\nabla I(x,y)| = \sqrt{I_x(x,y)^2 + I_y(x,y)^2} \tag{3.6}$$

Alternatively,

$$|\nabla I(x,y)| \approx |I_x(x,y)| + |I_y(x,y)| \tag{3.7}$$

The image intensity gradient direction $\theta(x,y)$ is given by Equation 3.8.

$$\theta(x,y) = \arctan\left(\frac{I_y(x,y)}{I_x(x,y)}\right) \tag{3.8}$$

3.2 Snake Finding Algorithm

The snake finding algorithm is an iterative process and can be described as a greedy search in the space of potential splines. The initial snake is generated either through user interaction or using an image processing algorithm which can localize the desired feature. Preferably, the initial snake should be located in the proximity of the feature of interest. The snake finding algorithm progresses by updating each snaxel with a member within its $M \times M$ neighborhood that minimizes the snake energy function. The algorithm terminates when a predefined stopping criteria is met. The algorithm is summarized in Algorithm 1.

There are two typical choices for the stopping criterion. The first option is to halt the snake finding algorithm after a preset number of iterations. The second option is to cease

Algorithm 1 Snake Algorithm

1. Initialize a snake $S^0 = \{p_1^0, \ldots, p_N^0\}$ in the proximity of the desired image feature.
2. At iteration T+1, set p_i^{T+1} to a member within the $M \times M$ neighborhood of p_i^T that minimizes $E(p_i)$ for all p_i.
3. Repeat step 2 until one of the stopping criteria is met.

iterating if the number of snaxels moved in the previous iteration is less than a specified threshold (i.e. the snake has settled sufficiently into a local minimum).

3.3 Kalman Snake

A common problem with the standard snake finding algorithm is its reliance on image forces during the process of capturing the true contour. The result is that the contour tracking is often ine ective if the snake ever finds itself in a homogeneous region. An obvious solution is to estimate the location of the boundary and then position the snake in the proximity of the boundary. The Kalman filter [21] is a recursive estimator which estimates the current state of the system based purely on the previous state and the current measurement(s). The recursive nature of the Kalman filter makes it a very efficient technique for object tracking. The Kalman filter assumes that the dynamic system can be modeled using a linear set of equations presented using the following customary notation:

$$\begin{aligned} x_t &= Ax_{t-1} + w \\ z_t &= Hx_t + v \end{aligned} \tag{3.9}$$

where x_t and z_t are, respectively, the estimated state of the system and the measurement generated at time t. The state transition matrix A describes the relationship between the current state and the previous state, whereas the measurement matrix H relates the current

measurement to the current state. The terms w and v denote the noise components of the model which account for any inaccuracies in the underlying model assumptions and the imprecision inherent in real-life measurements. The process noise w and measurement noise v are modeled as zero-mean Gaussian random variables with standard deviation q and r, respectively. The entities A, H, w, and v are assumed to be time-invariant, however, certain domain-specific knowledge may encourage the use of their time-dependent variants (i.e. $A(t)$, $H(t)$, $w(t)$, $v(t)$).

The model used for contour tracking assumes that the object boundary can be approximated as an ellipse. This restriction simplifies the state representation since all that is required are the parameters (a, b, c, d, f, g) of the general quadratic equation (Equation 3.10) for an ellipse. The constraints on the ellipse parameters are given in Table 3.1.

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0 \tag{3.10}$$

$$\Delta = \begin{matrix} a & b & d \\ b & c & f \\ d & f & g \end{matrix} \qquad J = \begin{matrix} a & b \\ b & c \end{matrix} \qquad I = a + c$$

$$\Delta \neq 0 \quad J > 0 \quad \Delta / I < 0$$

Table 3.1: Defining Properties of an Ellipse

In addition to the ellipse parameters, the changes in the ellipse parameters are also included in the state equation. These are required in order to model the movement of the contour. Hence, the state variable x_t can be represented as a 12-dimensional vector. The state transition matrix A is easily derived from the set of equations relating each state variable in one frame with the corresponding state variable in the subsequent frame (i.e. $a_t = a_{t-1} + \delta a_{t-1}$ and $\delta a_t = \delta a_{t-1}$). The state variable and state transition matrix are shown in Table 3.2.

The ellipse parameters are the measured variables; hence, a 6-dimensional vector is used to represent the measurement variable z_t. The relationship between the measurement variable

and the state variable is given by the measurement matrix H. Both matrices are shown in Table 3.3.

$$x_t = [a, b, c, d, f, g, \delta a, \delta b, \delta c, \delta d, \delta f, \delta g]^T$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Table 3.2: State Variable x_t and the State Transition Matrix A

$$z_t = [a, b, c, d, f, g]^T$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 3.3: Measurement Variable z_t and the Measurement Matrix H

The Kalman snake requires an initial state x_0 and state covariance matrix P_0 to commence the recursive snake finding algorithm. The Contour Tracker treats the user-generated initial snake as the starting state x_0. The state covariance matrix P_0 is set with values determined from experimental data generated using a standard snake. The choice of x_0 is reasonable since the user will presumably initialize the snake finding algorithm with a high-quality preliminary snake.

The Kalman snake begins with the a priori prediction of the current state x_t^- based on the previous state x_{t-1}. A measurement z_t is generated by running the snake finding algorithm, on frame f_t, starting with the ellipse described by the parameters in x_{t-1}. The next step is to combine x_t^- and z_t to update the Kalman filter's a posteriori estimate x_t. After the

Kalman filter has generated the current state x_t, the snake algorithm is performed on frame f_t using x_t as the initial snake. The steps are repeated iteratively until all frames have been processed. The Kalman snake algorithm is summarized in Algorithm 2.

Algorithm 2 Kalman Snake Algorithm

1. Initialize the a posteriori state x_0 and a posteriori state covariance P_0.
2. Predict the a priori estimate x_t^- based on the previous state x_{t-1}.
3. Generate a measurement z_t by performing the snake algorithm starting with x_{t-1}.
4. Update the filter's a posteriori estimate x_t by combining x_t^- and z_t.
5. Run the snake finding algorithm using x_t.
6. Return to step 2 if there are more frames.

3.4 Gradient Vector Flow

The Kalman snake is a valid approach to resolving the initialization problem; however, it has its own set of weaknesses. Its main drawback is that it is designed to model strictly linear systems. A contour whose movement is not representative of a linear model will not benefit from using the Kalman snake algorithm described above. [22] propose the gradient vector flow (GVF) field as an alternative solution to the initialization problem. The GVF targets the external energy term of the snake equation. In traditional snakes, computation of the external energy term typically involves standard edge detection algorithms that use the gradient or Laplacian operators. The GVF expands upon these approaches by using the edge maps to detect the changes in the gradient. The GVF field is the vector field $v[u(x,y), v(x,y)]$ that minimizes the energy functional defined in Equation 3.11.

$$E(\mathbf{v}) = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2 dxdy \tag{3.11}$$

where f is an edge map (or gradient magnitude map) of the original gray-scale image $I(x,y)$ (i.e. $f(x,y) = |\nabla I(x,y)|^2$). In Equation 3.11, one can notice that the second term dominates

in areas of high discontinuities (associated with large values of $|\nabla f|$), whereas the first term dominates in homogeneous regions. The result is a gradual dispersion of the gradient vectors away from object boundaries. This property of the GVF assists the snake algorithm in capturing the contour because it extends beyond the distance a orded by standard gradient-based vector fields.

3.5 Droplet Parameter Extraction

Given the ellipse parameters of the droplet boundary from each frame, certain features of the droplet contour can be computed using the equations in Table 3.4. The point (x_0, y_0) is the center of the ellipse, a' and b' are the radii of the major and minor axis respectively, and θ is the angle of rotation of the major axis.

$$x_0 = \frac{cd - bf}{b^2 - ac} \quad y_0 = \frac{af - bd}{b^2 - ac}$$

$$\theta = \frac{1}{2}\arctan\left(\frac{2b}{c-a}\right)$$

$$a' = \sqrt{\frac{2(af^2 + cd^2 + gb^2 + 2bdf - acg)}{(b^2 - ac)\left[(c-a)\sqrt{1 + \frac{4b^2}{(a-c)^2}} - (c+a)\right]}}$$

$$b' = \sqrt{\frac{2(af^2 + cd^2 + gb^2 + 2bdf - acg)}{(b^2 - ac)\left[(a-c)\sqrt{1 + \frac{4b^2}{(a-c)^2}} - (c+a)\right]}}$$

Table 3.4: Ellipse Feature Equations

Chapter 4

Description of Experimental Setup

The nanostructured surface used for the current study is comprised of Silicon nanorods and was prepared by a process of *glancing angle deposition* (GLAD). A detailed description of the GLAD process is beyond the scope of this paper, but is discussed in [5, 15, 16, 23]. Briefly, in the GLAD process, an RCA-1 cleaned p-type Silicon (100) substrate is installed in an electron beam evaporation system (Torr International, Inc.), in such a way that the normal to the substrate surface makes an angle $\theta = 86°$ with the incoming vapor (Figure 4.1). The evaporation system is then pumped down to a base pressure of around $10^{-6} - 10^{-7}$ Torr. During the vapor deposition, a stepper motor rotates the substrate azimuthally about its surface normal at an angular speed of $0.05 rev/sec$. The film thickness and deposition rate are monitored using a quartz crystal microbalance (QCM). The Silicon nanorods prepared

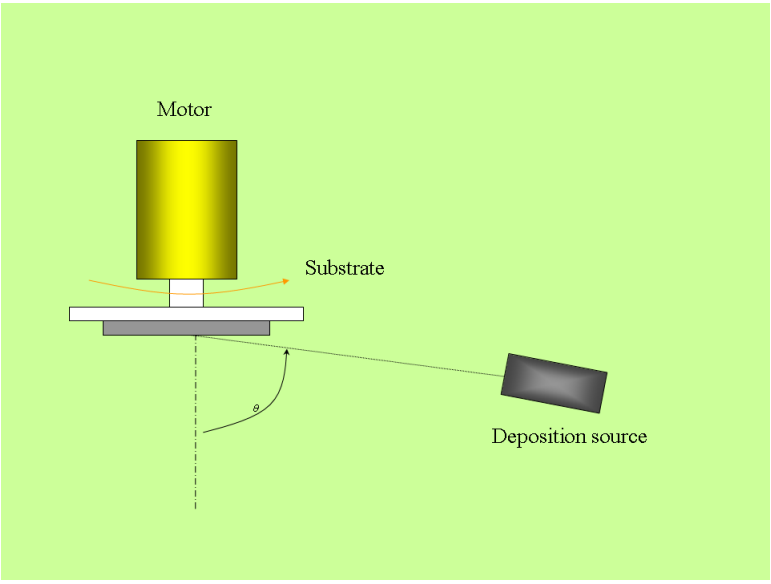

Figure 4.1: A sketch showing the glancing angle deposition method for growing nanorods.

through the GLAD process were vertically aligned on the substrate. The spatial distribution of the nanorods was observed to be statistically uniform. This is shown by the top-view and cross-sectional view scanning electron microscope (SEM) images in Figure 4.2. The inter-nanorod separation distances and the heights of the nanorods are easily controllable. The morphological parameters obtained from the SEM images are as follows: the average diameter of the nanorods at the top of the nanostructured surface $d_t = 133 \pm 34nm$, the average inter-nanorod separation distance $L = 280 \pm 106nm$, and the average nanorod height $h = 1106 \pm 104nm$. For the purpose of measurement and analysis of the dynamics of wettability, a $3\mu L$ water droplet was dispensed through a $250\mu L$ Hamilton syringe onto a sample of the freshly prepared nanostructured surface. A very slow approach speed was utilized in order to minimize the e ect of the impact of the water droplet on the surface. The video of the dynamics of the spreading droplet was taken with a fast CCD camera (Pyramid Imaging Inc.) and recorded using the XCAP software (EPIX, Inc.). During the entire experiment, the temperature was maintained at $\sim 25°C$ and the humidity was maintained at $\sim 23\%$.

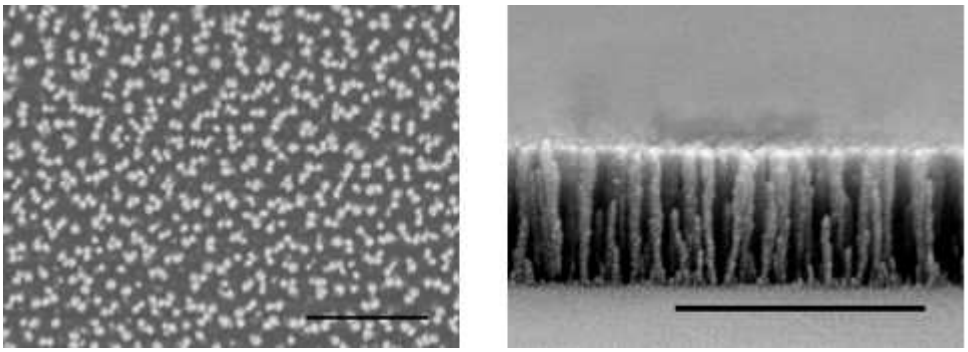

Figure 4.2: Top-view and cross-sectional view of Silicon nanorods grown by glancing angle deposition (GLAD). The deposition rate was $0.2nm/sec$. The scale bar is $2\mu m$.

The Contour Tracker application was developed using the Open Source Computer Vision (OpenCV) Library (introduced by Intel Corporation), in conjunction with Microsoft's Visual C++ .Net architecture. The gradient vector flow algorithm was implemented using the code developed by [22][1].

[1]available at http://iacl.ece.jhu.edu/projects/gvf/

The application employs the snake algorithm to track a single contour within a video or sequence of frames. The current version features the two aforementioned common variations of snake tracking algorithms: the standard snake developed by [9] and the Kalman snake developed by [20]. Four additional external energy operators are included for added functionality. They are the Sobel operator, the Canny operator, edge detection with embedded confidence [14], and the gradient vector flow field [22].

The outputs of the program are a tab-delimited text file specifying the ellipse parameters (a through g) and droplet parameters (center point, major and minor axes, and rotation angle of the major axis) for each frame as well as the video displaying the estimated location of the contour. Figure 4.3 through 4.6 are screenshots of the Contour Tracker. Figure 4.3 is the main screen of the application which displays the current frame being processed. Figure 4.4 is the tracking parameters dialog screen showing the available tracking options. Figure 4.5 shows an example of an initialized contour. Figure 4.6 shows the captured droplet boundary.

Figure 4.3: *Contour Tracker* main screen.

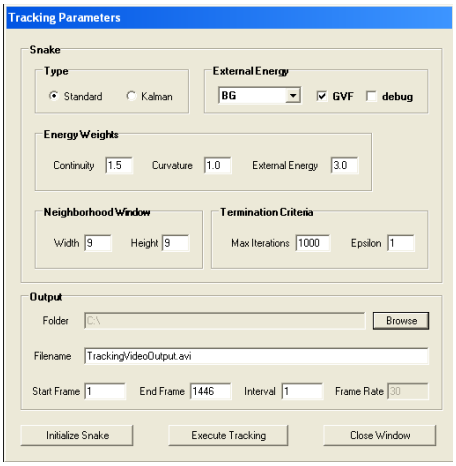

Figure 4.4: Tracking parameters dialog.

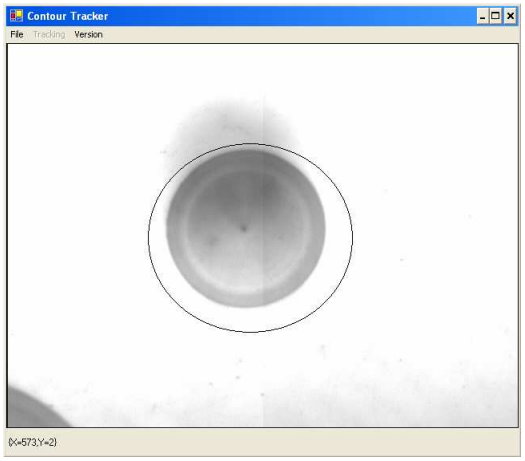

Figure 4.5: Image of an initialized contour.

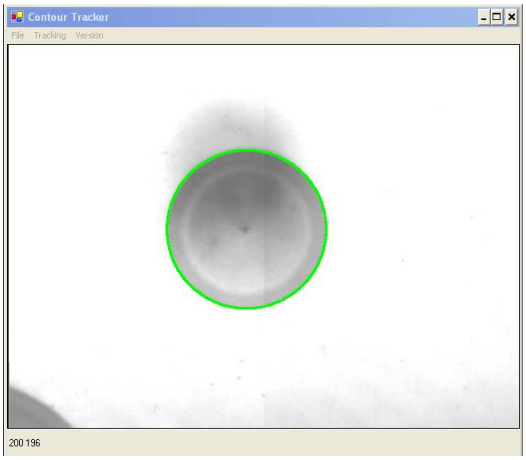

Figure 4.6: Image of a snake capturing the water droplet boundary.

Chapter 5

Results and Analysis

Figure 5.1 shows a comparison of the precursor radius R_p obtained through manual measurement and determined by the *Contour Tracker*. We can observe that manual measurement provides a limited number of data points, some of which are unacceptable, whereas the contour tracking program provides a smooth plot. Linearization of the two curves using a log-log scale (Figure 5.1(b)) for values of $t < 3.5$ seconds yields two di erent values for the scaling exponent : 0.136 resulting from manual measurement and 0.146 resulting from the contour tracking program. We trust the result from the contour tracking program since the graphical user interface (GUI) allows us to examine the evolution of the tracked contour on a per-frame basis. Any tracking errors will be evident in the GUI. Table 5.1 compares the precursor radius at three di erent time instances as obtained from manual measurements and the *Contour Tracker*.

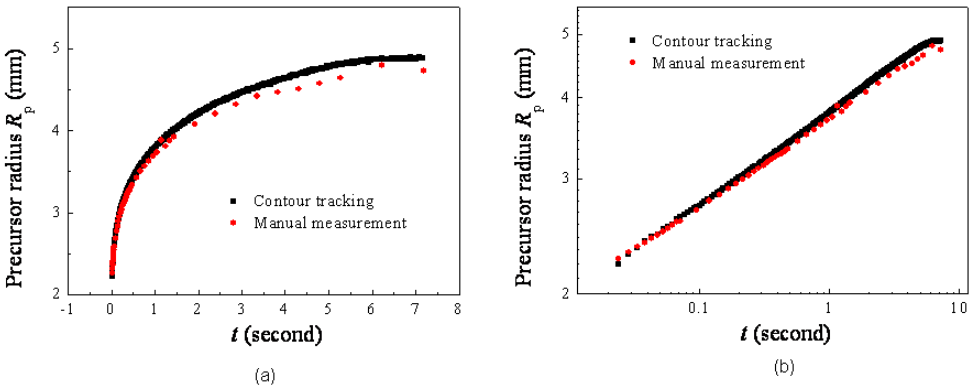

Figure 5.1: Comparison of the evolution of the precursor radius R_p over time t calculated from manual measurement and using the *Contour Tracker* on a (a) linear scale and (b) log-log scale.

Time (second)	0.053	1.055	5.273
Manual (mm)	2.491	3.731	4.639
Contour Tracker	2.513	3.829	4.819
Percent di erence (%)	0.9	2.6	3.7

Table 5.1: Comparison of the precursor radius R_p obtained from manual measurements and the *Contour Tracker* at 3 time instances.

Figure 5.2 plots the evolution of the instantaneous droplet spreading speed, which is obtained simply via the temporal di erentiation of the curves in Figure 5.1 followed by a 5-adjacent point smoothing procedure. For small values of t, the values of the instantaneous droplet spreading speed resulting from manual measurement and the contour tracking program di er; this can be attributed to measurement accuracy. However, for large values of t, we observe that the speed from the contour tracking is very noisy. The fact is, for large values of t, the change in the precursor radius from one frame to the next is very small (less than one pixel for a single frame interval). However, since the current version of the contour tracking program only outputs integer values for the contour position it is not well-suited from a metrological standpoint for cases where the changes in contour position are very small. Subpixel interpolation techniques are called for under these circumstances.

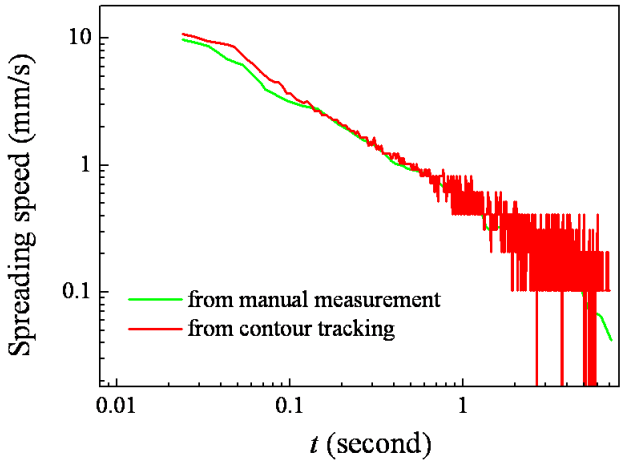

Figure 5.2: The instantaneous spreading speed of the precursor radius obtained from the temporal di erentiation of the curves in Figure 5.1 followed by a 5-adjacent point smoothing procedure.

Chapter 6

Concluding Remarks and Future Work

Dynamic wettability measurement and analysis of a solid nanostructured surface is a topic of great importance in a variety of problem areas. The dynamic wettability analysis is performed by measuring the evolution of the precursor (outer rim) contour of the water droplet as it spreads on a solid nanostructured surface. We developed a computer program based on the snake active contour model which is capable of precisely tracking the precursor contour of a spreading water droplet in a high frame-rate video. Experiments show reasonable agreement between the results of the tracking program and those obtained via manual measurement. However, the current version of the contour tracking program is only able to detect the outer rim or precursor contour of the water droplet. This alone is not enough to describe the dynamics of the spreading behavior of the droplet. A fair comparison of the dynamics of water droplet spreading on a nanostructured surface to that on a flat, rough or porous surface requires information about the contact line and hence the length between the contact line and the precursor as a function of time. This calls for being able to track the evolution of the contact line. The contour tracking algorithm also needs to provide sub-pixel accuracy in cases where the di erences in contour pixel locations from one frame to the next are very small. These research tasks will be pursued in our future work.

Part II

A Multiscale Background Method for Static Object Detection [1]

[1] S. Cheng, X. Luo, S. M. Bhandarkar. 2006. To be submitted to 2007 IEEE Workshop on Applications of Computer Vision.

Chapter 7

Introduction

The detection of stationary objects is a primary objective for many security-based surveillance systems present in train stations, airports, and other highly-populated locations. Immediate detection of suspicious packages is vital to the safety of innocent citizens in the current age of terrorists and primitive home-made explosive devices. On a more trivial scale, static object detection can be used to signal incidents of left-luggage at transportation hubs as well as detecting illegally parked vehicles on roadsides. This paper proposes a multiscale background approach to static object detection.

Background modeling (sometimes referred to as figure-ground discrimination) is the process of subtracting the background from an image in order to analyze the actions and behaviors of foreground objects. Background identification is an important step in computer vision applications, particularly in the field of surveillance and monitoring. A common obstacle known as the *sleeping person problem* is encountered by most surveillance systems. The sleeping person phenomenon occurs when a foreground object remains stationary for an extended period of time. After a while, the foreground object will merge into the background image because the background model will begin to associate the object's pixel as the estimated background pixel. The merging time is dependent on the scale of the model. Consequently, static objects in the scene can be detected by exploiting the sleeping person problem and using a multi-model, or multi-scale, architecture.

The following chapters outline the proposed multi-scale approach to static object detection. Chapter 8 explains the multiple Gaussian mixture background model and the subse-

quent multiscale strategy. Chapter 9 describes the experiment setup and results. Chapter 10 summarizes the results and suggests possible extensions to the research.

Chapter 8

Background Modeling

In computer vision, there are two common approaches to modeling the background image of a real-time scene. The first method is to use a single color value for each background pixel. Typical univalue color schemes use temporal averaging [8] or temporal median filtering [12] to calculate the background pixel values. Although these methods are effective in scenes where objects are in constant motion and the background is visible for the majority of the time, they are not well-suited for busy scenes in which there are many moving objects, especially if the objects are slow-moving. Another drawback of single color models is that bimodal background pixels cannot be modeled using a single color value [18]. The second approach combats these shortcomings by utilizing a multi-color scheme to model the background image. The mixture of Gaussians (MoG) background model [18] is an adaptive background mixture model in which each background pixel value is modeled as a set of Gaussian distributions. The MoG performs well under various conditions such as different equipment, indoor/outdoor environments, and a range of lighting scenarios [18].

8.1 Background Image Assumptions

In order to distinguish the background image from moving objects, there are assumptions that must be accepted. The first assumption is that background pixel values tend to persist for longer durations than pixel values of foreground objects. This assumption is naturally true for scenes involving a few fast-moving objects. However, a scene containing many objects (i.e. a busy scene) is a difficult scenario for less-sophisticated background models. The second assumption relates to the busy scene problem and states that background pixel

values recur more frequently than foreground pixel values. This situation occurs when foreground objects periodically occlude the background image. The proposed background model, which is explained in more depth in the following section, incorporates these assumptions into its background pixel weighting system.

8.2 Multiple Gaussian Mixture (MGM) Background Model

The multiple Gaussian mixture (MGM) background model, developed by [11] and inspired by the MoG, is a multi-color, statistical approach to background extraction. The MGM model incorporates k Gaussian distributions for each pixel of the image $I_{x,y}$. From this point forward, all terms are assumed to be associated with a particular pixel so the x and y subscripts will be omitted. Each distribution, or color cluster, χ_i is characterized by the following attributes:

1. μ_i: Mean.
2. σ_i^2: Variance.
3. N_i: Weight.
4. tl_i: Time that χ_i was last updated.
5. n_i: Number of image intensity values that have matched χ_i in the current time-slice.

The background updating process employs a two-stage approach. The initial stage tallies the statistical properties of the pixel value for each of the k distributions. The image intensity I_t in frame t is compared to each of the k color clusters. I_t matches cluster χ_i if I_t falls within 2.5 standard deviations of χ_i's mean (i.e. $\mu_i - 2.5\sigma_i \leq I_t \leq \mu_i + 2.5\sigma_i$). A match results in the updating of μ_i and σ_i^2 according to Equations 8.1 and 8.2, respectively.

$$\mu_i = \mu_i + \frac{1}{L}(I_t - \mu_i) \tag{8.1}$$

$$\sigma_i^2 = \sigma_i^2 + \frac{1}{L}[(I_t - \mu_i)^2 - \sigma_i^2] \tag{8.2}$$

where L is an integer representing the inverse of the learning rate. The use of an integer speeds up the algorithm by avoiding floating-point arithmetic. If I_t does not match any of the clusters, then the cluster with the lowest weighting N is replaced with a new cluster χ_j where $\mu_j = I_t$, $\sigma_j^2 = \sigma_0^2$, $N_j = 1$, $n_j = 1$, and $tl_j = t$. The variance σ_0^2 is initialized to a high value because it is assigned to newly created clusters.

The second phase of the background updating model occurs every F frames and involves updating the cluster weights. Each color cluster is assigned a weight N_i representing the likelihood that χ_i corresponds to the actual background pixel value. N_i takes into account the two important assumptions of background pixels mentioned in Section 8.1, the amount of time a certain color persists (duration) and its recurrence frequency. At each interval F, the cluster weights are updated according to the duration and recurrence frequency weight update rules described by Equations 8.3 and 8.4, respectively. The recurrence frequency affects the cluster weight only if the cluster has been sufficiently represented (i.e. $n_i > \delta$ where δ is an integer) during the recent time slice.

$$N_i = N_i + \begin{cases} F, & \text{if } n_i > F/2 \\ n_i, & \text{otherwise} \end{cases} \tag{8.3}$$

$$N_i = \begin{cases} N_i + F/2, & \text{if } n_i > \delta \text{ and } t - tl_i > 2F \\ N_i, & \text{otherwise} \end{cases} \tag{8.4}$$

All clusters satisfying the condition $N_i > N_{max}/3$ where $N_{max} = max(N_i)$, are deemed to represent a background pixel value. Once the background pixels have been determined for the current frame, if $N_{max} > 1.25\Delta$, then each N_i is scaled by a factor of 4/5. This is necessary because if N is unbounded, then a cluster with a relatively large N would dominate the system making it difficult for other clusters to be considered as part of the background. Any cluster with $N = 0$ or $t - tl > \Delta$ is deleted. The full background model updating procedure is described in Algorithm 3.

Algorithm 3 Background Model Updating

1. If I_t matches χ_i, then update μ_i and σ_i^2 using Equations 8.1 and 8.2. Set $n_i = n_i + 1$.
2. If I_t does not match χ_i, then replace the cluster with the lowest N value with a new cluster initialized with $\mu = I$

period of time. B_t includes all objects that have been motionless for a period $T_s > t$ where $T_s \approx N_i/frame\ rate$. It can be shown that if $i \leq j$, then $B_j \subseteq B_i$. In other words, objects in the coarser scale background image will also be present in the finer scale background image.

A system that utilizes two background models, B_i and B_j where $i < j$, is able to detect objects that have been static for at least i seconds (i.e. $T_s \geq i$) because after approximately i seconds, the object will merge into the background of B_i, but remain in the foreground of B_j. The location of the object can be determined by generating a difference background image $DB_i = B_i - B_j$.

Chapter 9

Experiment and Results

The multiscale background updating model is tested on two unique video sequences. Both scenarios are situated in an indoor environment: a living room and a train terminal [1]. In both experiments, the four model parameters are set to $k = 4$, $L = 1024$, $F = 60$, and $\delta = 20$. In other words, four Gaussian distributions are used for each pixel, the learning rate is set to $1/1024$, weight updates occur every $2s$ (60 frames), and the recurrence threshold is set to 20 updates.

9.1 Living Room

For the living room scenario, two background models, B_{30} and B_{60}, are trained for approximately 5400 frames (∼180s). This ensures that both models are sufficiently trained since $\Delta_{B_{30}} = 2700$ and $\Delta_{B_{60}} = 5400$. The training period consists of a static background scene with minimal illumination changes. Immediately after the training period, a person enters the scene, places a laptop bag on the ground, and proceeds to leave the camera's field of view. The laptop bag remains in the scene for approximately 3750 frames (∼125s), at which time a person retrieves the bag returning the scene to its original state.

Figure 9.1 shows the state of the overall system at frame 5753 (∼192s). Figure 9.1(a) and (b) show the background image generated by B_{30} and B_{60}, respectively. It can be seen that the laptop bag is still present in the foreground of both background models. Figure 9.1(c) is the difference background image DB_{30} and Figure 9.1(d) shows the original image from the

[1] The train terminal video is provided by the PETS2006 workshop with the support and collaboration of the British Transport Police and Network Rail.
http://www.cvg.rdg.ac.uk/PETS2006/index.html

video. Since DB_{30} shows no image, it can be concluded that the object has been static for less than 30s and hence, no warning is issued.

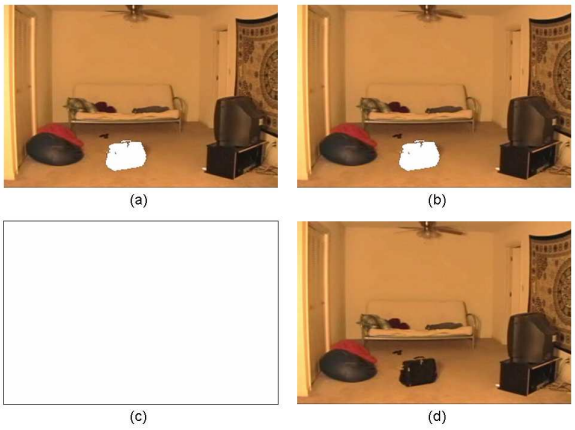

Figure 9.1: Images of frame 5753. (a) Background image B_{30}. (b) Background image B_{60}. (c) Difference background image $DB_{30} = B_{30} - B_{60}$. (d) Monitor view.

Approximately 976 frames (∼32.5s) after the bag is placed in the scene, the object begins to merge into the background of B_{30}. Figure 9.2 is a snapshot of the system at frame 6729. Figure 9.2(a) and (b) show the states of B_{30} and B_{60}, respectively. The object is present in B_{30}, but does not appear in B_{60} implying that the object has been static for more than 30s, but less than 60s. This is confirmed in DB_{30} shown in Figure 9.2(c). The 30s warning indicator is shown in Figure 9.2(d) as a yellow bounding box around the object.

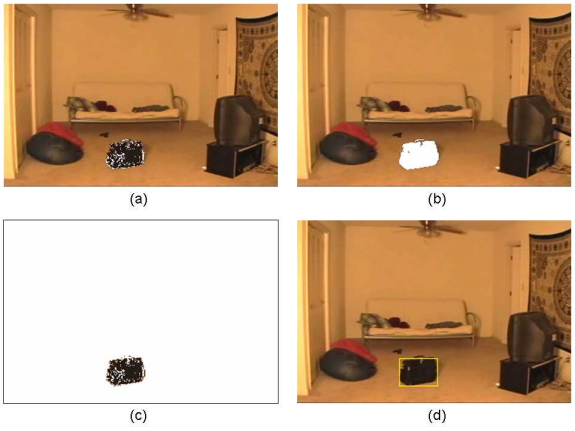

Figure 9.2: Images of frame 6729. (a) Background image B_{30}. (b) Background image B_{60}. (c) Difference background image $DB_{30} = B_{30} - B_{60}$. (d) Monitor view with warning indicator (shown as a yellow bounding box).

9.2 Train Terminal

Unlike the controlled, quiet scene of the living room, the train terminal video is a more complicated scenario. Three background models, B_5, B_{15}, and B_{20}, are instantiated to supply the system with both a warning period ($5 \leq T_s < 15$) and a higher priority alarm signal ($T_s \geq 15$). The flexibility of the multiscale approach is shown by using this two-stage alert system.

Throughout the entire video, various people pass through the scene simulating the movement of real-life commuters. The training period for this experiment is set to around 1800 frames (∼20s), which is equivalent to the largest Δ of the three background models. Approximately 1700 frames (∼57s) into the video, a man enters the scene with a long carrying case. He lingers around the target area for approximately 30s, at which time he leans the case up against the railing and proceeds to leave the scene. For the remainder of the video, the object remains in the same location while commuters continue navigating through the camera's field of view, occasionally hiding portions of the object.

Figures 9.3 through 9.5 show key frames of the video at three separate time instances. For each of the figures, (a) through (c) show the three background models B_5, B_{15}, and B_{20}, respectively. Also, (d) and (e) are the difference background images DB_5 and DB_{15}, and (f) is the image shown on the monitor display.

Figure 9.3 is a snapshot of frame 2792 in which the man has left the carrying case against the railing and begins to exit the scene. The object has been static for less than 5 seconds since DB_5 does not identify any connected components large enough to classify as an object. The foreground (black) pixels of DB_5 can be attributed to the non-uniform distribution of the object pixels in the scene. For example, the majority of movement occurs within the walking area of the train terminal and because of the camera angle, the quieter portions of the scene (i.e. pixels in the upper area of the image) receive less pixel value variation than their busier counterparts.

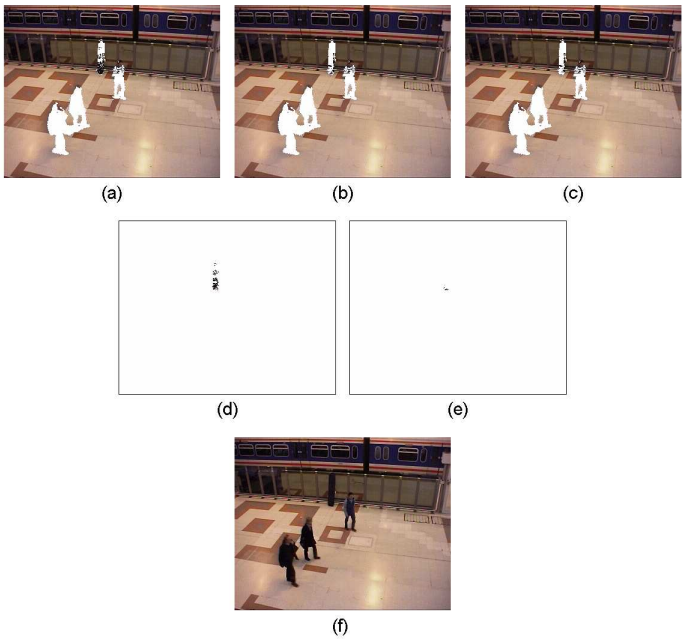

Figure 9.3: Images of frame 2792. (a) Background image B_5. (b) Background image B_{15}. (c) Background image B_{20}. (d) Difference background image $DB_5 = B_5 - B_{15}$. (e) Difference background image $DB_{15} = B_{15} - B_{20}$. (f) Monitor view.

Figure 9.4 shows the state of the system at frame 2946. This is approximately 5 seconds after the object was left unattended, which agrees with the theoretical time when the static object should merge into B_5. The static object appears in DB_5 indicating that the object has been stationary for greater than 5 seconds. The system notifies the monitoring party by displaying a yellow bounding box around the object in the system's monitor view.

Once the object remains idle for more than 15 seconds, an alarm is triggered and the proper authorities are notified automatically so that they can attend to the potential danger immediately. Figure 9.5 shows the video at frame 3219, which is approximately 15 seconds after the initial time the object was left. The alarm is displayed as a red bounding box around the object in the system's monitor view.

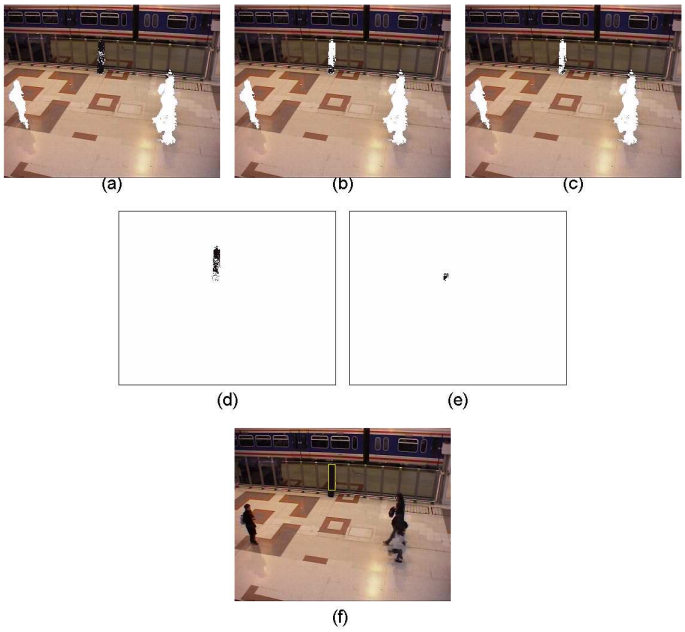

Figure 9.4: Images of frame 2946. (a) Background image B_5. (b) Background image B_{15}. (c) Background image B_{20}. (d) Difference background image $DB_5 = B_5 - B_{15}$. (e) Difference background image $DB_{15} = B_{15} - B_{20}$. (f) Monitor view with warning indicator (shown as a yellow bounding box).

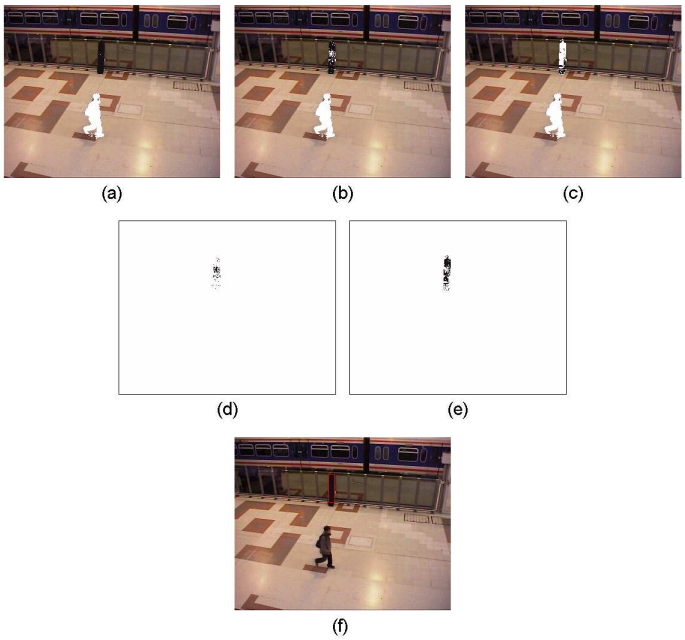

Figure 9.5: Images of frame 3219. (a) Background image B_5. (b) Background image B_{15}. (c) Background image B_{20}. (d) Difference background image $DB_5 = B_5 - B_{15}$. (e) Difference background image $DB_{15} = B_{15} - B_{20}$. (f) Monitor view with alarm indicator (shown as a red bounding box).

Chapter 10

Concluding Remarks and Future Work

The results of the study show that using a multiscale background scheme is a valid approach to dynamic scene analysis. The system was able to signal a warning when an object remained stationary for the specified amount of time. Although the system performed reasonably well, a few weaknesses were exposed while performing the experiments. The first problem concerns the accuracy of the system when the background models are not sufficiently trained (i.e. $F_t \ll \Delta$, where F_t is the length of the training period in frames). Consequently, if the models do not have the correct estimate of the background image, foreground objects will merge into the background sooner than expected.

Another issue related to the training problem is the situation in which portions of the same object merge into the background at different rates. This problem could result in undersized connected components which will affect the object detection process. Another consequence of the merging problem is the difficulty of getting a sharp transition between the warning and alarm stages.

A more accurate assessment of the proposed scheme can be obtained by integrating the model into a real-time, online monitoring system. In an online situation, the processing speed of the entire system is of utmost concern. Fortunately, the multiscale approach is well-suited for a parallel computer architecture where each model can be managed independently using a multi-processor system.

The sleeping person problem has a counterpart known as the waking person problem. The waking person scenario occurs when an object that originates in the background is moved. A missing objects detection feature could be implemented by utilizing the effects of the waking

person problem. This function is useful in theft prevention systems. It could also be used to find misplaced items in a crowded room. The drawbacks, improvements and suggestions discussed in this chapter will be investigated in future studies.

Bibliography

[1] M. Apel-Paz and A. Marmur. Spreading of liquids on rough surfaces. *Colloids and Surfaces A: Physiochemical and Engineering Aspects*, 146(1):273–279, 1999.

[2] L. Bacri and F. Brochard-Wyart. Droplet suction on porous media. *European Physical Journal E - Soft Matter*, 3:87–97, 2000.

[3] A. M. Cazabat and M. A. C. Stuart. Dynamics of wetting on smooth and rough surfaces. *Colloid and Polymer Science*, 74(1):69–75, 1987.

[4] A. A. Darhuber, S. M. Troian, and W. W. Reisner. Dynamics of capillary spreading along hydrophilic microstripes. *Physical Review E*, 64(3):031603, 2001.

[5] J. G. Fan, X. J. Tang, and Y. P. Zhao. Water contact angles of vertically aligned si nanorod arrays. *Nanotechnology*, 15:501–504, 2004.

[6] P. G. De Gennes. Wetting: statics and dynamics. *Reviews of Modern Physics*, 57(3): 827–863, 1985.

[7] S. Gerdes, A. M. Cazabat, G. Ström, and F. Tiberg. Effect of surface structure on the spreading of a pdms droplet. *Langmuir*, 14(24):7052–7057, 1998.

[8] S. Kamijo. Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):108–118, 2000.

[9] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[10] Kenneth K. S. Lau, José Bico, Kenneth B. K. Teo, Manish Chhowalla, Gehan A. J. Amaratunga, William I. Milne, Gareth H. McKinley, and Karen K. Gleason. Superhydrophobic carbon nanotube forests. *Nanoletters*, 3(12):1701–1705, 2003.

[11] Xinzhi Luo and Suchendra M. Bhandarkar. Real-time and robust background updating for video surveillance and monitoring. *Lecture Notes in Computer Science*, 3656:1226 – 1233, 2005.

[12] M. Massey and W. Bender. Salient stills: Process and practice. *IBM Systems Journal*, 35(3&4):557–573, 1996.

[13] G. McHale, N. J. Shirtcliffe, S. Aqil, C. C. Perry, and M. I. Newton. Topography driven spreading. *Physical Review Letters*, 93(3):036102, 2004.

[14] Peter Meer and Bogdan Georgescu. Edge detection with embedded confidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1351–1365, 2001.

[15] K. Robbie and M. J. Brett. Sculptured thin films and glancing angle deposition: Growth mechanics and applications. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 15(3):1460–1465, 1997.

[16] J. C. Sit, D. Vick, K. Robbie, and M. J. Brett. Thin film microstructural control using glancing angle deposition by sputtering. *Journal of Materials Research*, 14:1197–1199, 1999.

[17] V. M. Starov, S. A. Zhdanov, S. R. Kosvintsev, V. D. Sobolev, and M. G. Velarde. Spreading of liquid drops over porous substrates. *Advances in Colloid and Interface Science*, 104:123–158, 2003.

[18] Chris Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition*, 2:246–252, 1999.

[19] L. H. Tanner. The spreading of silicone oil drops on horizontal surfaces. *Journal of Physics D: Applied Physics*, 12(9):1473–1484, 1979.

[20] Demetri Terzopoulos and Richard Szeliski. Tracking with kalman snakes. In *Active Vision*, pages 3–20. MIT Press, 1993.

[21] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[22] Chenyang Xu and Jerry L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

[23] Y. P. Zhao, D. X. Ye, G. C. Wang, and T. M. Lu. Designing nanostructures by glancing angle deposition. *Proceedings of the 48th SPIE Conference*, 5219:59–73, 2003.