Comprehensive 3-d Change Detection Using Volumetric Appearance Modeling

by

Thomas B. Pollard

B.A., Mathematics, Pomona College 2002

Thesis

Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

May 2009

Abstract of "Comprehensive 3-d Change Detection Using Volumetric Appearance Modeling," by Thomas B. Pollard, Ph.D., Brown University, May 2009

The problem of detecting changes in images taken by a stationary camera has been well studied and many algorithms now exist which perform robustly in real-world applications. However no comprehensive solution exists for the analogous "3-d" change detection problem, where the camera taking the images is allowed to move and the scene is not planar. This situation is common in a growing number of applications, especially in aerial surveillance where the recent explosion of available aerial imagery has made manual examination by analysts infeasible. This thesis presents the first comprehensive solution to the 3-d change detection problem, using a new framework called volumetric appearance modeling (VAM). This approach can manage the complications of unknown and sometimes changing world surfaces by maintaining a 3-d voxel-based model, where probability distributions for surface occupancy and image appearance are stored in each voxel. These distributions are continuously updated as new images are received using an adaptive learning procedure. VAM is demonstrated to perform well on several aerial image sequences taken from helicopter from a wide range of viewpoints. The core VAM framework is extended to additionally deal with the variable lighting and haze conditions present in satellite imagery. The success of this complete framework is demonstrated on selected scenes of Baghdad, Iraq exhibiting a challenging range of viewpoints, lighting directions, and haze.

This dissertation by Thomas B. Pollard is accepted in its present form by
the Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of
Doctor of Philosophy

Date_____ _____

Joseph L. Mundy, Director

Recommended to the Graduate Council

Date_____ _____

David B. Cooper, Reader

Date_____ _____

Donald McClure, Reader

Approved by the Graduate Council

Date_____ _____

Sheila Bonde

Dean of the Graduate School and Research

The Vita of Thomas B. Pollard

Thomas Benjamin Pollard was born in Juneau, Alaska on February 4, 1980. He graduated from Pomona College in Claremont, California in May 2002 *Magna Cum Laude* with a B.A. in mathematics. The following fall he began his graduate studies at the applied math department of Brown University. Over the next few years his interests shifted towards the more applied fields of computer vision and graphics. In 2004 he began working with Prof. Joseph L. Mundy in the Brown engineering department on developing a state of the art algorithm for change detection in 3-d scenes. The volumetric appearance modeling framework presented in this thesis is the culmination of that effort.

Acknowledgments

This thesis is the culmination of four years of research into change detection that began with simple planar background modeling experiments and evolved into a powerful and novel voxel-based framework called volumetric appearance modeling (VAM). None of this would have been possible without the guidance and patience of my advisor Prof. Joseph Mundy, who provided frequent and valuable advice over the course of the project. I am also grateful for the funding provided by the National Science Foundation VIGRE program and the National Geospatial Intelligence Agency for my research. The VIGRE support in particular came at a crucial time in my early research years and allowed me to work uninterrupted on developing the VAM framework which would later be mature enough to support additional funding.

I would also like to thank the other professors in the Brown LEMS group: Prof. David Cooper, Prof. Benjamin Kimia, and Prof. Gabriel Taubin for listening to my ramblings at our weekly lab meetings and providing additional valuable advice. I am equally indebted to the other graduate students in LEMS who took over development of the next generation software implementation of VAM in early 2008 so that I could write this thesis and graduate! Lastly I want to thank the countless friends in Providence who made the last six years so memorable, and my parents for supporting my graduate school ambitions.

Contents

Acknowledgments **v**

1 Introduction **1**

1.1 Outline 9

2 History of Change Detection **10**

2.1 2-d Change Detection 11

2.1.1 Early Algorithms 11

2.1.2 Stauffer-Grimson and Extensions 12

2.2 3-d Change Detection 14

2.2.1 Registration Techniques 14

2.2.2 Manual Site Modeling 19

2.2.3 Motion Detection 20

2.2.4 Other Techniques 21

2.3 Conclusions 22

3 Volumetric Appearance Modeling (VAM) **24**

3.1 Framework 26

3.2 Online Updating 28

3.2.1 Updating the Occupancy Probabilities 29

3.2.2 Updating the Appearance Models 32

3.3 Detecting Change 34

3.4 Rendering Synthetic Images 35

3.5 Convergence . . . 35
3.6 Comparison with Space Carving . . . 38
3.7 Summary . . . 39

4 Experiments **41**
4.1 Example Sequences . . . 41
4.1.1 Plasticville . . . 42
4.1.2 Steeple Street . . . 46
4.1.3 Post Office . . . 51
4.1.4 Capitol Building . . . 51
4.2 Analysis of Errors . . . 53
4.2.1 Misalignment Error . . . 56
4.2.2 Non-Lambertian Surfaces . . . 58
4.2.3 Sampling Deficiency . . . 64
4.3 Conclusions . . . 66

5 VAM for Satellite Imagery **67**
5.1 Local-Lighting Appearance Model . . . 68
5.2 Intensity Normalization . . . 69
5.3 Experiments . . . 71
5.3.1 Normalization Accuracy . . . 74
5.3.2 Hiafa St. . . . 75
5.3.3 Orchard . . . 75
5.3.4 U.S. Embassy . . . 81
5.3.5 Tigris River . . . 81
5.3.6 Karkh . . . 83
5.3.7 Predicting Appearance . . . 86
5.4 Conclusions . . . 87

6 Using VAM with Arbitrary Data Sources **89**
6.1 General Updating . . . 90

6.2 LIDAR Updating 91
6.2.1 DEM LIDAR 93
6.2.2 Point-Cloud LIDAR 94
6.3 Conclusions 96

7 Implementation **97**
7.1 Plane Stack 98
7.2 Supervoxels 102
7.3 Additional Implementation Issues 103
7.3.1 Voxel/Image Resolution 103
7.3.2 Normalization Implementation 105
7.3.3 Miscellaneous Speedups 106
7.4 bvxm 107

8 Conclusions **108**

List of Figures

1.1 Sample views of a scene in Baghdad, Iraq from a set of 32 satellite images taken over 5 years, demonstrating the range of viewpoints, lighting conditions, and haze typically encountered in aerial imagery from multiple collections. 2

1.2 Two satellite images of the same scene taken 2 years apart, illustrating the types of changes of interest in this thesis. Interesting physical changes to the scene like structures being built and cars on the road should be detected while differences caused by a new viewpoint and lighting should not be. Physical "change" can sometimes be an ambiguous concept, as illustrated by the shifting dirt on the road and new permutation of cars in the parking lot. What is the "normal" appearance of these regions if they are different in every image? 4

1.3 A summary of the types of uninteresting changes which occur in images due to variability in the imaging conditions. These changes can be broken down into four categories: camera viewpoint, lighting conditions, weather/seasons, and normal variation of the background environment. Some of these changes such as weather and lighting occur only in long term collection scenarios taken over months or years, while variations caused by viewpoint and moving vegetation can occur even when all images are taken at the same time of day. The algorithm presented in this thesis attempts to model all of this variation in its background model with the exception of weather/seasonal effects, which will not be considered. Any variation not caused by one of these categories should be detected as change. 5

1.4 A preview of the volumetric appearance modeling (VAM) framework introduced in this thesis. The world is filled with voxels which model the appearance of the pixels they project into in each image. Voxels which see consistent intensities have higher probability of being part of the world surface. The right rendering shows voxels with high surface probability after training on 30 images. 8

2.1 An example from the original Stauffer-Grimson paper [41]. (left) a new frame, (middle) an image composed of the means of the most probable Gaussians in the background model, (right) the detected foreground pixels . 12

2.2 Some example recovered background mosaics used for change detection by Ren et al. (top) and Mittal and Huttenlocher (bottom). 17

2.3 An example site model from Huertas and Nevatia [23]. Manually constructed geometry indicated by white lines models most of the 3-d structure of the scene. 18

2.4 A screenshot of the RADIUS site modeling application showing the types of structures that can be manually created including roads, buildings, and water ways (Copyright SRI International 1994). 19

2.5 Change detection results from Del Frate et al. [14]. (Left and Middle) classification maps from 2002 and 2003 respectively where black: asphalted surface, white: buildings, dark gray: bare soil, and light gray: vegetation. (Right) detected changes. 22

3.1 An example showing a voxel array drawn on top of a scene. Note that this voxel array does not contain all of the 3-d structure in the scene. The top of the tallest building lies outside the volume and so its background appearance cannot be learned. 25

3.2 Voxel notation. X is voxel on the world surface $\mathcal{S}$ lying on the ray $R(X)$ and projecting into image I at pixel y. However V_R is the voxel that actually produced color I_y. 26

3.3 A sample mixture of Gaussians distribution with $K = 3$ modes. 27

4.1 A volume rendering of the recovered voxel geometry from the plasticville scene after training on 40 images. White voxels have higher probability. Though this still image doesn't do it justice, the recovered geometry is close to the true 3-d surface except in areas of uniform texture like the parking lot where there is a layer of low probability voxels which together have high probability. 44

4.2 Examples from VAM trained on the plasticville model town sequence. (left column) a new unseen image, (middle column) the expected appearance from this viewpoint, (right column) the changes detected with VAM. . . . 45

4.3 For uniformly colored surfaces such as roads imaged from a limited range of viewpoints the shape of the true surface is ambiguous. For the camera angles in this example a surface above or below the actual road could all have produced the observed images. When VAM is run on the images, the recovered voxel geometry will be a layer of voxels with low probability which together add up to high probability. 46

4.4 Change detection results for VAM and a planar algorithm applied to the plasticville image sequence. A) an unseen image, B) hand-marked ground truth changes on this image, C) results from a planar change detection algorithm, D) VAM change detection results. The planar algorithm has significant errors do to unmodeled geometry, while VAM produces scattered small misregistration errors. 47
4.5 Change detection ROC curves for the model town sequence. Results are given for different viewing angles, measured from the normal to the ground plane. Voxel algorithm gives superior all around performance, with a greater difference at larger angles. 48
4.6 Examples from VAM trained on the Steeple St. sequence. (top) a new unseen image, (middle) the expected appearance from this viewpoint, (bottom) the changes detected with VAM. 49
4.7 An example of the difficulties VAM has with non-Lambertian surfaces, where false detections are made on the metal roof from a certain viewing direction. (top) a new unseen image, (middle) the expected appearance from this viewpoint, (bottom) the changes detected with VAM. 50
4.8 ROC curves comparing the performance of VAM vs the planar change detection algorithm on the Steeple St. sequence. Violations of the constant color assumption due to non-lambertian surfaces cause more false positives in this sequence than for the plasticville sequence, but the voxel algorithm still outperforms the planar algorithm significantly. 51
4.9 Two examples from VAM trained on the post office sequence. Cars driving around the building are correctly detected as change in both examples. The specularities on the roof that occur in the right image cause additional false detections. 52
4.10 ROC curves comparing the performance of VAM vs the planar change detection algorithm on the post office sequence. 53

4.11 Examples from VAM and the planar algorithm trained on the capital building sequence. (top) a new unseen image, (middle) the detected changes using the planar algorithm, (bottom) the changes detected with VAM. The only true changes in this scene are the vehicles passing in the far street (inside dashed boxes) which are easily detected by VAM without many false alarms on the capitol. The planar algorithm fails to cope with the extreme viewpoint shift over the sequence. 54
4.12 ROC curves comparing the performance of VAM vs the planar change detection algorithm on the capitol building sequence. 55
4.13 A volume rendering of the recovered voxel geometry from the capital building sequence. White voxels have higher probability. 55
4.14 Histograms of observed intensities in neighborhoods around pixels on sharp edges. Samples were obtained by taking all pixels within 1 pixel of a line along the indicated edges. The solid graph for each example shows the observed intensities along each edge. The dashed graphs show the observed intensites in the flat regions to either side of the edges for comparison. The histograms on the edges are fairly uniformly spread over the area between the border intensities, though there may be dips due to low sampling. These histograms indicate the level of variation one could expect to see in a voxel on one of these edges. 57
4.15 False detections are made on this specular metal roof when viewed from certain directions. The errors are especially bad when first observing the specularity (middle), however after training on a wider range of viewpoints the specularity has higher probability (bottom). 59
4.16 Sample observations of several synthetic specular surfaces viewed from a circular arc overhead. Each bottom figure shows a surface seen from all points on a viewing hemisphere, with the black circle indicating a special viewing path. The graphs above show the observed intensities along this viewing path. Depending on the surface orientation and specular radius the variation in intensities may be extreme or non-existent. 60

4.17 Examples of the Phong model with α=8, 16, 64 and 256. 61
4.18 Variables in the Phong model. 61
4.19 The density function from equation 4.2.4 plotted for various values of α. . . 62
4.20 Density functions for mixtures of Gaussians trained on simulated Phong observations for different values of α. All learned distributions have a peak at the Lambertian color 0.2 and the maximum specular value which varies. The specular peak for $\alpha = 64$ is at 0.8 but has so little weight in the mixture that it is almost unnoticable. 63
4.21 ROC curves comparing the fraction of correctly detected changes (uniform distribution) to the fraction of false alarms (Phong distribution). 64
4.22 Change detection ROC curves for VAM applied to the plasticville sequence with different numbers of training images. 20 images is sufficient to produce good change detection, with more images providing marginal improvement. 65

5.1 The space of lighting directions is discretized with each bin containing an independent mixture of Gaussians appearance model. 69
5.2 A plot of the range of lighting and camera directions present in the Baghdad satellite image collection. Views are shown from the side and top-down with red lines for camera directions and blue lines for lighting directions. The camera directions are fairly uniformly distributed across the top of the sphere. Since the satellite is sun-synchronized all images were taken at around 7:45 AM and as a result the lighting directions are concentrated in a narrow strip to the southeast. 72
5.3 Intensity differences between an image and its normalized version after correcting for artificially added haze. Errors are quite low even after training on just one image in both average and worst-case errors for a set of 8 images. (Error is expressed in percent) 74

5.4 Example detections for the Hiafa St. sequence. Top row shows before and after images of the scene, while bottom row shows change detection results from VAM and joint probability. Most of the detections in VAM are vehicles with some interesting structural changes in the upper right corner. The joint probability technique makes significant errors on the sides of the tall buildings since it doesn't model 3-d structure. 76
5.5 ROC curves for the Hiafa St. sequence. VAM is able to detect about half of the vehicle pixels in the scene with an acceptable false alarm rate. The joint probability technique makes 2-3 times as many false detections for any reasonable percentage of correct detections. 77
5.6 Some interesting structural changes in the Hiafa St. scene detected in a training image. 77
5.7 A volume rendering of the recovered voxel geometry from the Hiafa St. sequence. White voxels have higher probability. 78
5.8 False positive detections in the Hiafa St. sequence on sun-facing walls. . . . 78
5.9 Example detections for the orchard sequence. Top row shows before and after images of the scene, while bottom row shows change detection results from VAM and joint probability. The changes detected by VAM on the right side of the street contain most of the true change to the scene. There are very few true changes on the left side of the street where the joint probability produces many false detections. 79
5.10 A volume rendering of the recovered voxel geometry from the orchard sequence. White voxels have higher probability. The orchard region has many voxels with relatively high occupancy probability in the empty air above the trees, unlike the neighboring urban area. 80
5.11 ROC curves for the orchard sequence. 81
5.12 Example detections for the embassy sequence. Top row shows before and after images of the scene, while bottom row shows change detection results from VAM. The trailers in the right side construction lot and scattered changes in the left courtyard are all detected with few false alarms. 82

5.13 Sample images of the Tigris river over the years 2002-2007. 83
5.14 Errors in learning the Tigris River. The highly variable appearance of the river causes large errors on August 2005 in the middle of the image sequence. By the end of the sequence in May 2007 enough variability has been introduced into the river's appearance models that they are unable to identify all of the debris floating by. 84
5.15 Progress at a construction site in the Karkh neighborhood 2002-2007. . . . 84
5.16 Changes detected by VAM in a new image of the Karkh construction site compared to the previous image. In this constantly changing scene it is difficult to determine what the correct classification of foreground and background should be. Here VAM identifies some small buildings and piles of dirt as change, while not making many false detections on the far right building. However it misses two important changes to the scene circled in white. 85
5.17 Two examples using a predicted appearance model for unseen lighting categories. Figures are an unseen image (a), its expected appearance (b), and changes detected (in black) for 20° (c) and 40° (d) gaps in trained lighting. Change detection for the 20° predicted appearances is highly accurate, however for the low lighting angle in the bottom example the change detection for 40° has some false alarms from shadows which do not appear in either neighboring appearance model. 87

6.1 Two common forms of LIDAR are the raw point cloud output (left) and a processed digital elevation map (DEM) where brighter pixel intensities indicate taller structures (right). Left image copyright Ambercore Software, Inc. 92

7.1 Camera rays intersecting a voxel volume organized in planes parallel to the z-axis. In (A) the camera position is such that all rays intersect the planes in top-down order, making occlusion calculations easy. In (B) with the side camera, rays above the middle ray intersect the stack in one order while rays below intersect in the reverse order. This complication, in addition to the need to pay close attention to the order of voxels in each plane, makes occlusion computations harder to implement in a plane stack framework. . . 98

7.2 Voxels can be grouped into larger "supervoxel" blocks which are loaded from and saved to disk together. 102

7.3 Assignment of voxels to camera rays for different image resolutions. Camera rays are shown as black lines while voxels are shaded grey according to which ray they belong. The resulting voxel-rays are a) of unit thickness when pixel spacing is just right, b) too thick when pixel spacing is too large, and c) discontinuous when pixel spacing is too small. 104

Chapter 1

Introduction

The problem of **change detection** in images is a fundamental and well-studied problem in computer vision. In its simplest formulation the change detection problem is, given two images of the same scene, to determine which pixels appear different between the images. Many modern change detection algorithms can obtain very accurate results by using many images of the same scene to learn the most common appearance of each pixel over the sequence. Pixels in a new image not close enough to this learned normal appearance are determined to be changes. Such algorithms are called **background modeling** techniques because they learn what the normal "background" appearance of the scene should look like when there are no other objects present. By using many images to learn the normal appearance of a scene, a background modeling algorithm can say with greater confidence which pixels should be considered abnormal changes, and thus obtain higher change detection accuracy.

The difficulty of the background modeling problem is proportional to the variation encountered in a "typical" set of images of the scene. If the camera taking the images is stationary, the lighting constant, and foreground objects rare then the background can be represented as the running average of the image intensities. However in practice these properties rarely hold. For example in a highway surveillance video there may be trees blowing in the wind, lighting changing as clouds move overhead, and the road may more often than not be occluded by foreground vehicles. In such a situation the amount of variation makes learning the background appearance a much harder problem. Still

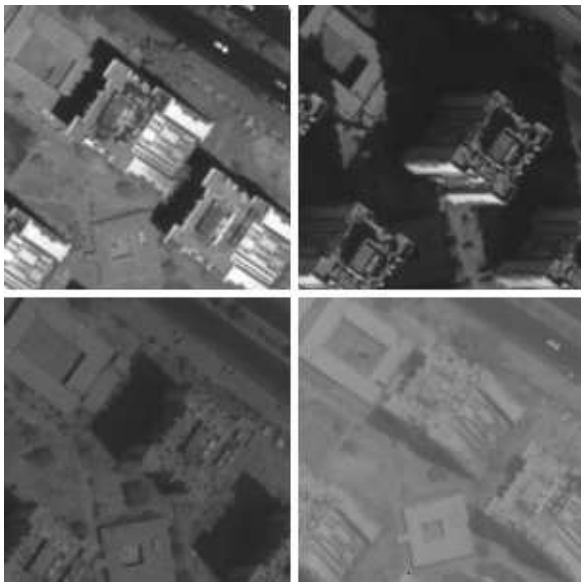

Figure 1.1: Sample views of a scene in Baghdad, Iraq from a set of 32 satellite images taken over 5 years, demonstrating the range of viewpoints, lighting conditions, and haze typically encountered in aerial imagery from multiple collections.

there are many good change detection algorithms that perform well provided that the camera is stationary, even in the presence of a dynamic background or slowly changing illumination [41] [25]. These algorithms are widely used in surveillance systems as well as in preprocessing video for a variety of computer vision applications.

However there are many potential applications for change detection where the images are not taken from a stationary camera. The main example of this is aerial imagery, where images are usually collected from a large range of viewpoints and at different times of year. The variability introduced by viewing 3-d structures from arbitrary viewpoints and times is enormous, and images of the same scene can often look nothing alike (see figure 1.1). Current "2-d" image-based background modeling algorithms are inherently ill-suited for learning the 3-d background scene in such scenarios. What is needed are "3-d" background modeling and change detection algorithms which can cope with the variation introduced by arbitrary viewpoints. Change detection in this 3-d context is a largely unexplored problem in computer vision, yet the benefits of an accurate and robust system are boundless.

The most direct application of 3-d change detection is in military surveillance using

aerial imagery. Petabytes of imagery from satellites and UAVs are collected every day around the globe and must be analyzed in great detail by intelligence analysts to determine enemy activity. A system capable of highlighting areas of interest in these images would greatly reduce the analysis effort and decrease missed detections caused by human error. Border patrol is a related application that would benefit from fast and accurate detection of suspicious activity from UAV patrols.

A major non-military commercial application of change detection in satellite imagery is land cover classification. Analysis carried out in forestry, hydrology, agriculture, geology, and environmental protection is interested in changes in vegetation and soil over time. A 3-d change detection algorithm using satellite imagery could identify areas of changing land use more accurately than current methods.

A less obvious application is to industrial inspection systems. In a typical inspection system, a newly manufactured object is placed on a table and images/x-rays are taken from many viewpoints to identify any defects. Currently the images are compared to reference images from a "ground-truth" object, however this 2-d image comparison requires that the new images be taken from the exact viewpoint as the reference images which may be hard to control. Instead one could use a 3-d change detection algorithm to learn the structure and appearance of the "ground-truth" object and use this to detect changes in images of a newly manufactured object. Learning the full 3-d appearance of the object would be a much more accurate means of detecting manufacturing defects.

Since change detection is a common preprocessing step in a large number of computer vision problems such as tracking, object recognition, and activity analysis, a solution to the 3-d change detection problem means that many of these techniques can be transported from the 2-d domain to 3-d. For example, an algorithm for classifying vehicles in a road-side video which requires a segmentation of the car from the background as a pre-processing step could potentially be done from a traffic helicopter using a 3-d change detection scheme.

Yet there is still no complete and accurate solution to the 3-d change detection problem. One reason for the lack of an existing solution is the sheer amount of technical hurdles involved with getting accurate geo-registration and and camera calibration for aerial im-

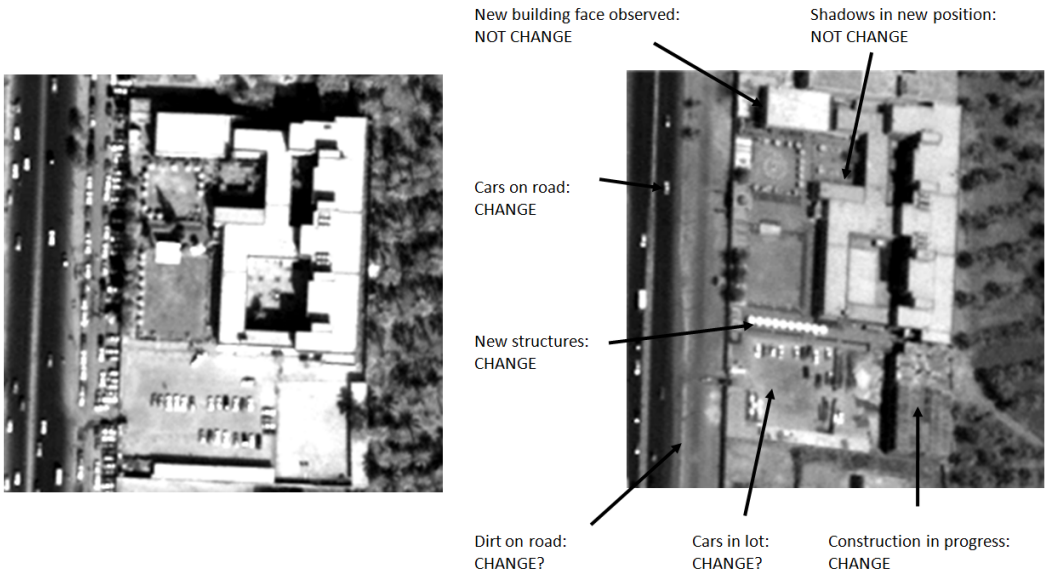

Figure 1.2: Two satellite images of the same scene taken 2 years apart, illustrating the types of changes of interest in this thesis. Interesting physical changes to the scene like structures being built and cars on the road should be detected while differences caused by a new viewpoint and lighting should not be. Physical "change" can sometimes be an ambiguous concept, as illustrated by the shifting dirt on the road and new permutation of cars in the parking lot. What is the "normal" appearance of these regions if they are different in every image?

ages. Furthermore, once the images are in a state where they can be used, there is still a very challenging vision problem to be solved. As seen in figure 1.1, variability in viewpoint, lighting, and atmospheric conditions typically results in images of the same scene which appear dramatically different from each other. These differences make it very difficult to detect interesting physical changes in the scene while not detecting uninteresting changes in pixel appearance due to viewpoint and lighting variability.

The definition of "interesting" here is somewhat subjective, but it should include physical changes to the scene that a human observer could distinguish. An example of the types of interesting changes that one would like to detect is given in figure 1.2. Physical changes such as vehicles on the road and new structures being built are of interest, while previously occluded building faces and moving shadows are not. Attempting to classify the dirt on the roads and the cars in the parking lot raises some interesting philosophical questions,

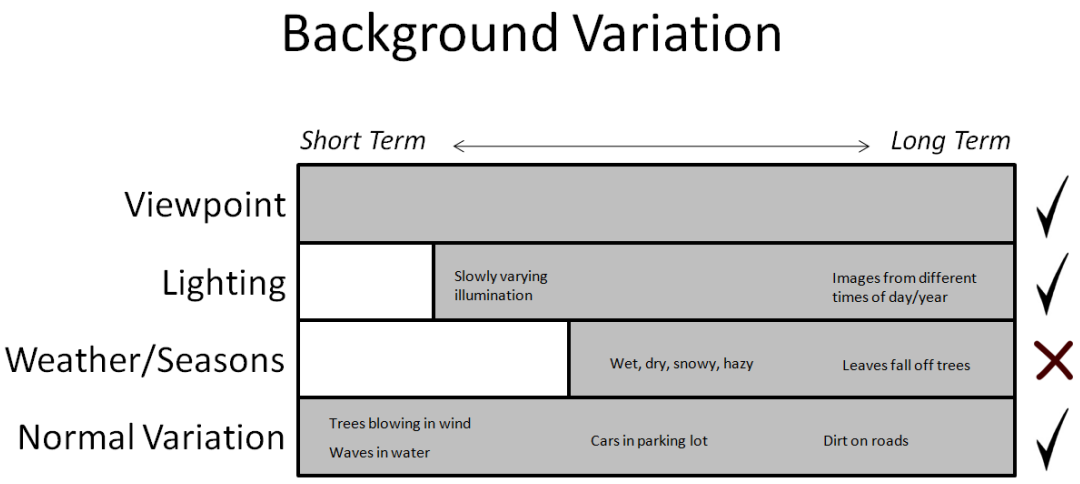

Figure 1.3: A summary of the types of uninteresting changes which occur in images due to variability in the imaging conditions. These changes can be broken down into four categories: camera viewpoint, lighting conditions, weather/seasons, and normal variation of the background environment. Some of these changes such as weather and lighting occur only in long term collection scenarios taken over months or years, while variations caused by viewpoint and moving vegetation can occur even when all images are taken at the same time of day. The algorithm presented in this thesis attempts to model all of this variation in its background model with the exception of weather/seasonal effects, which will not be considered. Any variation not caused by one of these categories should be detected as change.

since they are physical changes yet occur frequently enough that they are somehow "normal". The diagram in figure 1.3 attempts to resolve any ambiguity as to what kinds of variation should be considered background and what should be considered change. Pixel variation caused by differences in viewpoint and lighting should not be considered changes nor should normal physical variation in the scene. Normal physical variation is defined here to be reoccurring physical events intrinsic to part of the scene such as trees blowing in the wind in short term sequences or cars shifting in parking lots in longer term sequences. Any pixel variability not caused by any of the categories in figure 1.3 should be classified as change.

This thesis presents the first comprehensive solution to the 3-d change detection problem, able to distinguish interesting physical changes from those due to variability in the imaging conditions. The focus here is on detecting changes in aerial imagery from either satellite or aircraft, as this is the most common situation where 3-d change detection would be useful. However the core algorithm introduced in chapter 3 is very general and also applies to ground level change detection with a moving camera. The development of the framework presented in this thesis is guided by several requirements to guarantee that it would be usable under very general conditions. The key requirements are:

- The algorithm must be able to automatically detect all interesting changes to a scene. Interesting changes are defined to be physical changes to the scene which are not attributed to imaging variability as defined in figure 1.3. To be fully automatic the algorithm cannot require excessive work from the human user such as manual construction of 3-d geometry.

- Images of the scene can be taken from cameras with arbitrary but known position and orientation. It is assumed that all images are associated with a known camera model with sub-pixel reprojection error, which is either a standard 3x4 projective camera matrix or a cubic rational polynomial camera [18] for the case of satellite imagery. Aerial images from commercial or military sources typically come with such camera models which vary widely in accuracy. Automatic correction of these imperfect cameras is non-trivial but is a well-studied area of research, and will not

be discussed in this thesis.

- Images can be taken at any time during daylight hours and with variable amounts of haze, but no other weather effects (i.e. clouds, snow) may be present. This is a fair assumption for aerial imagery captured by satellite and airplane at different times. It is additionally assumed that the sun is the only light source and that the direction of light is known, calculated from standard solar equations and typically provided in satellite metadata.

- The algorithm must work when all images are taken in grayscale. If color imagery is provided then the color information should be used, however the algorithm should work in the absence of color. Typically color data comes at the expense of image resolution so many real world systems capture in panchromatic rather than RGB or multi-spectral. The algorithm presented in this thesis will only consider the case of grayscale imagery but can be extended to use color.

- The imagery may or may not be from a video source. No assumptions about frame-to-frame continuity can be used.

- The system should be able to use images taken from different sources together. For instance, appearance learned from satellite imagery should be able to detect changes in a aerial video sequence.

- Perhaps most importantly, the learning of the background appearance must be done **online**. An estimator of some quantity is said to be online if the current estimate can be incrementally refined with each new observation without reference to prior observations. In the context of change detection this means that the background model will need to be continuously updated as new images are taken. Being online is crucial for the algorithm to be usable in real world system since it must be able to automatically incorporate structural changes to the scene into the background model once they have been observed.

It is clear that to obtain comprehensive change detection results on images with arbitrary cameras a 3-d representation of the scene will be needed. Using a pre-computed

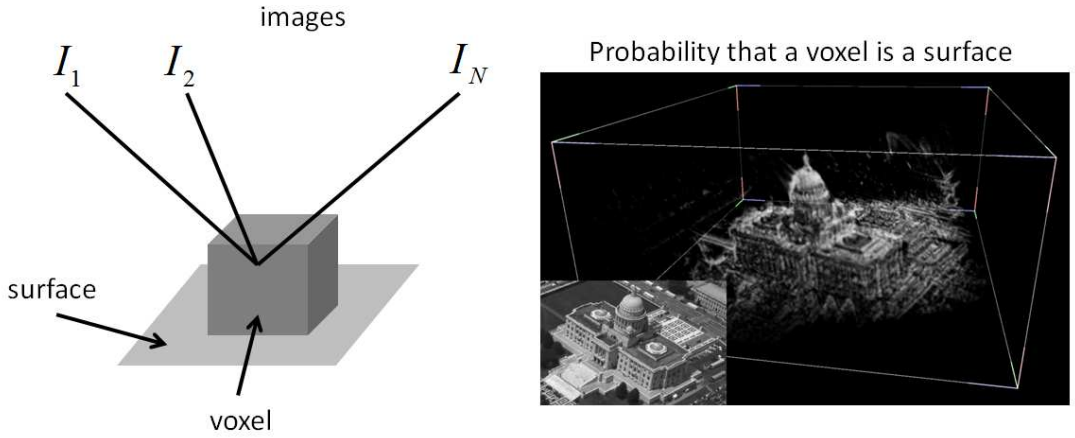

Figure 1.4: A preview of the volumetric appearance modeling (VAM) framework introduced in this thesis. The world is filled with voxels which model the appearance of the pixels they project into in each image. Voxels which see consistent intensities have higher probability of being part of the world surface. The right rendering shows voxels with high surface probability after training on 30 images.

3-d surface such as a mesh is an inadequate solution as there are many situations where the world surface may be changing over time (i.e. parked cars leaving, buildings being erected/demolished, etc.) and this surface will need to be recomputed frequently. This need to have an adaptable surface indicates that a volumetric model is more appropriate, so that hypotheses of all possible world surfaces can be maintained and reconsidered when evidence suggests a new surface has appeared.

This thesis proposes a new framework to change detection called **volumetric appearance modeling** (VAM). This approach can manage the complications of unknown and sometimes changing world surfaces by maintaining a 3-d voxel-based model, where probability distributions for surface occupancy and image appearance are stored in each voxel (see figure 1.4). The probability distributions at each voxel are continuously updated as new images are received. Lighting-dependent appearance models are used for scenes with variable illumination, and haze-type weather effects are factored out as a pre-processing step. Once the voxel model has been trained on sufficiently many images, it can be used to compute the probability of pixels in a new image to detect changes. The complete system is shown to provide accurate and comprehensive change detection on a variety of aerial

imagery. Furthermore the results from this thesis indicate that a voxel framework is an effective representation for integrating multiple sources of aerial data for many applications, not just change detection.

1.1 Outline

This thesis is organized as follows.

- Chapter 2 contains a comprehensive review of change detection literature, both 2-d and 3-d.
- Chapter 3 presents the main contribution of this thesis, the volumetric appearance modeling (VAM) framework and associated online updating equations. Equations for detecting changes and rendering synthetic images of the learned scene are also provided. Finally the convergence properties of the algorithm are investigated. This core theory was presented in the 2007 CVPR paper "Change Detection in a 3-d World" [32] by Pollard and Mundy.
- Chapter 4 presents extensive experiments of the framework described in chapter 3 on real world data sets. Sources of errors are analyzed.
- Chapter 5 describes how the framework can be extended to deal with the variable illumination and haze commonly encountered in satellite imagery. Additional experiments are performed on satellite images from Baghdad, Iraq.
- Chapter 6 generalizes the update framework to incorporate arbitrary types of data into the model. Updating with LIDAR data is given as an example.
- Chapter 7 discusses how the algorithms described in chapter 3 can be implemented efficiently. A fast and memory-efficient implementation is crucial for the system to be able to process an image in a reasonable amount of time.
- Chapter 8 concludes with a summary of the work done and possibilities for future research.

Chapter 2

History of Change Detection

Automated detection of changes in images is an old and well studied problem in computer vision. The earliest attempts at solving this problem used simple frame differencing, while modern solutions employ more and more sophisticated techniques from statistics and radiometry. This thesis builds upon many of the change detection techniques developed over the last few decades and a brief but thorough summary of relevant previous algorithms is a necessary starting point.

The vast majority of change detection algorithms assume that images are taken from a stationary camera at different times. This "2-d" change detection problem is the most well studied framework and many robust algorithms exist that solve this problem with high accuracy even in the presence of lighting changes. Limited work has been done to address the "3-d" change detection problem, where the camera viewpoint is allowed to change between images. Moreover, much of the work that has been done on 3-d change detection has been restricted to pan/tilt cameras or detecting only certain types of changes such as moving objects in video.

This chapter is laid out in two sections. The first section summarizes some classic algorithms from the 2-d change detection literature which are directly relevant to this thesis. The second section is a more extensive survey of existing 3-d change detection techniques and their current limitations.

2.1 2-d Change Detection

The literature on "2-d" change detection in the context of stationary video is vast and no attempt at a comprehensive review will be given here. A recent exhaustive survey of this literature is given in a 2005 paper by Radke et al. [34]. Instead this section is focuses mainly on the Stauffer-Grimson change detection algorithm [41] which is relevant to this thesis.

2.1.1 Early Algorithms

Some of the earliest approaches to change detection use simple frame differencing, i.e. thresholding the difference $|I_1(x, y) - I_2(x, y)|$ at each pixel in images I_1 and I_2. This approach is fast and easy to implement, and as such it is still widely used today. However there are many shortcomings of this approach. Firstly there is a problem of double detection of fast moving objects since there are differences in I_2 both where the objects used to be in I_1 and where they have moved to in I_2. There is also the issue that periodic motion of otherwise static objects, such as trees blowing in the wind, will be detected as change when for all practical purposes they should be classified as non-change. Some modern algorithms using frame differencing include [38], [40].

More sophisticated change detection techniques were later developed using statistical inference on pixel blocks. These algorithms typically perform a hypothesis test comparing the no-change null hypothesis H_0 to the change alternative hypothesis H_1. Aach et al [1] for example perform a significance test using a χ^2 statistic for the summed squared image difference in a pixel neighborhood. Black et al. [3] on the other hand instead classifies pixels into categories including 1) object or camera motion, 2) illumination phenomena, 3) specular reflection, and 4) "iconic changes" and pixels that do not fall in any of these categories are classified as outliers.

Another classical approach is to fit blocks of pixels in the images with polynomials and compare the fitted polynomials between images. Hsu et al. [21] use a likelihood ratio test of form:

Figure 2.1: An example from the original Stauffer-Grimson paper [41]. (left) a new frame, (middle) an image composed of the means of the most probable Gaussians in the background model, (right) the detected foreground pixels

$$\frac{\sigma_0^{2N}}{\sigma_1^N \sigma_2^N}$$

where N is the number of pixels in the block and σ_1, σ_2, σ_0 are the variances of the residuals from the fit using the blocks in image I_1, I_2, and both respectively.

These block-based techniques are an improvement upon simple frame differencing since they are less sensitive to small motions and, for some algorithms, illumination changes as well. However they still will make double detections on fast moving objects, since they only have two images to compare. When more images of the scene are available, a more complete picture of what the "background" should look like can be learned. Change detection algorithms that learn the normal appearance of the scene from many images and use it to identify abnormal changes are called background modeling techniques. Early attempts at background modeling [26],[24], [37] use a single Gaussian density at each pixel to model background intensity plus acquisition noise. These densities are updated with simple adaptive filters as new observations of the pixel are made. This framework doesn't suffer from the double detection problem of the previously mentioned techniques, however the single Gaussian cannot model the pixel variation present in many real world scenes. Situations such as frequently occluded surfaces (roads), slow moving objects, and vegetation blowing in the wind will cause errors in these kinds of systems.

2.1.2 Stauffer-Grimson and Extensions

It became clear at this point that a single mode distribution is not sufficient to model the background of these kinds of scenes. Friedman and Russell [15] conducted early

experiments on highway video with a mixture model of three predetermined Gaussian distributions corresponding to road color, shadow color, and vehicle colors. However the first fully general change detection algorithm using a multi-modal background model was presented in the well-known 1999 paper by Stauffer and Grimson [41]. Their algorithm uses a mixture of Gaussians density at each pixel to model a dynamic background. A mixture of Gaussians [10] is a probability density of form:

$$p(x|\Theta) = \sum_{k=1}^{K} p(k)p(x|k, \theta_k)$$

where the $p(x|k, \theta_k)$ are independent Gaussian distributions and the $p(k)$ are weights which sum to 1. The mixture of Gaussians is a widely used and well studied distribution for modeling multi-modal data. With a multi-modal background model, Stauffer and Grimson are able to easily learn the background of the kinds of complicated real world scenes mentioned above.

The updating of the mixture distribution at each pixel with a new image observation I_t is done using an online K-means approximation to the EM algorithm [9] as follows. The new pixel observation I_t is matched with the "closest" existing mode whose mean is within 2.5 standard deviations of the observation. The mean, variance, and weight of this mode are updated according to the following equations:

$$\begin{aligned} w_k^{t+1} &= (1-\alpha)w_k^t + \alpha \\ \mu_k^{t+1} &= (1-\rho)\mu_k^t + \rho I_t \\ (\sigma_k^{t+1})^2 &= (1-\rho)(\sigma_k^t)^2 + \rho(I_t - \mu_k^t)^2 \end{aligned} \tag{2.1.1}$$

where α and ρ are learning rates. If no existing mode is within 2.5 standard deviations to the observation a new mode is created, possibly destroying an existing low-weight mode. Heavier weighted modes are classified as background by a threshold and observations not matching to a background mode are considered change.

Because of its fast and robust performance on all types of realistic scenes, the Stauffer-Grimson algorithm is the standard technique for change detection. Most following papers

on 2-d change detection built on the mixture of Gaussians background model concept and either modified the update scheme or added further post-processing to handle more complicated scenes.

KaewTraKulPong and Bowden [25] use different update equations based on the online EM algorithm. These equations are almost the same as those used in Stauffer-Grimson except for the amount of modification to the mixture distribution allowed in an update step. Their framework allows for faster convergence to a stable background model and quicker adaptation to permanent changes in the scene. This improved framework is the basis for the surface appearance modeling component of this thesis, and is described in greater detail in chapter 3. KaewTraKulPong and Bowden also develop theory in their paper to detect and compensate for shadows of moving objects in color video. This extension is an improvement upon the Stauffer-Grimson algorithm, which cannot identify moving shadows from the objects that cast them.

2.2 3-d Change Detection

3-d change detection algorithms, where the camera is allowed to move between images, have evolved in parallel with the 2-d techniques over the years. There has been less work done by computer vision researchers on the 3-d problem than 2-d, both because it is harder and because the 2-d generality is sufficient for the majority of computer vision problems. Change detection in a 3-d context is of greater importance to the remote sensing community and much of the research done on it is in this literature. Though the developed 3-d change detection algorithms often draw from established 2-d techniques, the 3-d problem is difficult enough that often new and radical approaches must be developed.

2.2.1 Registration Techniques

One natural idea is to attempt to reduce the 3-d problem back to a 2-d problem by aligning images as near as possible and running a 2-d change detection algorithm. If the camera motion is mostly limited to pan/tilt or the scene is mostly planar then this approach has a chance of working reasonably well. However these algorithms will fail if there is

any 3-d relief between images, which is the case in the majority of applications. They will fail because no global image transformation exists which can line up 3-d geometry in images taken from wide viewpoints. This alignment of images is the so-called registration problem, and is a topic of considerable research in its own right.

To register two images I_1 and I_2, one needs to calculate a smooth mapping $T : I_1 \rightarrow I_2$ which takes a point in I_1 to its corresponding location in I_2. In general for aerial images such a mapping may only exist for points on a "ground plane" as points lying on 3-d structures may only appear in one of the two images, or require a "tearing" of the image to align pixels. However in the case of only camera rotation, a correspondence between all pixels can be achieved. This is because the rays through the camera center which form the images don't change when the camera rotates about its center. For most registration techniques the transformation is assumed to be a 3x3 homography matrix H [17] which operates on homogeneous image coordinates:

$$\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

which can account for the motion of all points on the ground plane assuming the images were taken with projective cameras. In the simpler case when the bottom row of H is [0 0 1] the homography is said to be affine and can accommodate translation, rotation, scaling, and skewing of the images, but not more complicated projective transformations.

For satellite imagery a popular registration technique is polynomial warping [33]. In this algorithm the mapping T is a polynomial transformation calculated from key point correspondences. In the case of a quadratic warp the transformation is:

$$T_x(x, y) = a + bx + cy + dxy + ex^2 + fy^2$$
$$T_y(x, y) = g + hx + iy + jxy + kx^2 + ly^2$$

This transformation allows for more bending of the image than a homography, but it is

still a global transformation of the whole image and cannot do local alignment of objects with 3-d relief between images.

Carlotto [6] uses a manual first-order polynomial warp (another name for an affine homography) to register satellite images and then compares the aligned images after compensating for global differences. He calculates a joint histogram between two images I_1 and I_2 based on corresponding pixel intensities and uses this histogram to form a conditional probability density $p(x_2|x_1)$ of a pixel intensity x_2 in I_2 given x_1 in I_1. This conditional probability is used to compute a forward prediction function:

$$f_{21}(x_1) = \int x_2 p(x_2|x_1) dx_2 \tag{2.2.1}$$

A similar backward prediction function $f_{12}(x_2)$ predicts the pixel intensity in I_1 given x_2. The forward and backward errors between the observed and predicted intensities in I_2 are:

$$\epsilon_F(i,j) = x_2(i,j) - f_{21}(x_1(i,j))$$
$$\epsilon_B(i,j) = x_1(i,j) - f_{12}(x_2(i,j))$$

which are thresholded to identify changes between the two images. These prediction equations are used to adjust for global differences between the two images such as those due to solar angle, sensor gain, and atmospheric transmission and scattering. This gives a more accurate result than just comparing the two images directly, however it still doesn't compensate for 3-d relief between images. Another weakness of this approach is the needed for a manual image registration which may be time consuming.

Other authors have done this registration automatically, though limited to the case of an affine transformation. Dai et al. [8] use strong edges such as river boundaries to compute affine transformations between images, but strong edges may not exist in many scenes. Habib et al. [16] present a semi-automatic affine registration technique using the Modified Iterated Hough Transform and detect changes in edge features, which are invariant to lighting changes. However an affine transformation is insufficient to correct for 3-d relief of structures in high-res imagery.

Figure 2.2: Some example recovered background mosaics used for change detection by Ren et al. (top) and Mittal and Huttenlocher (bottom).

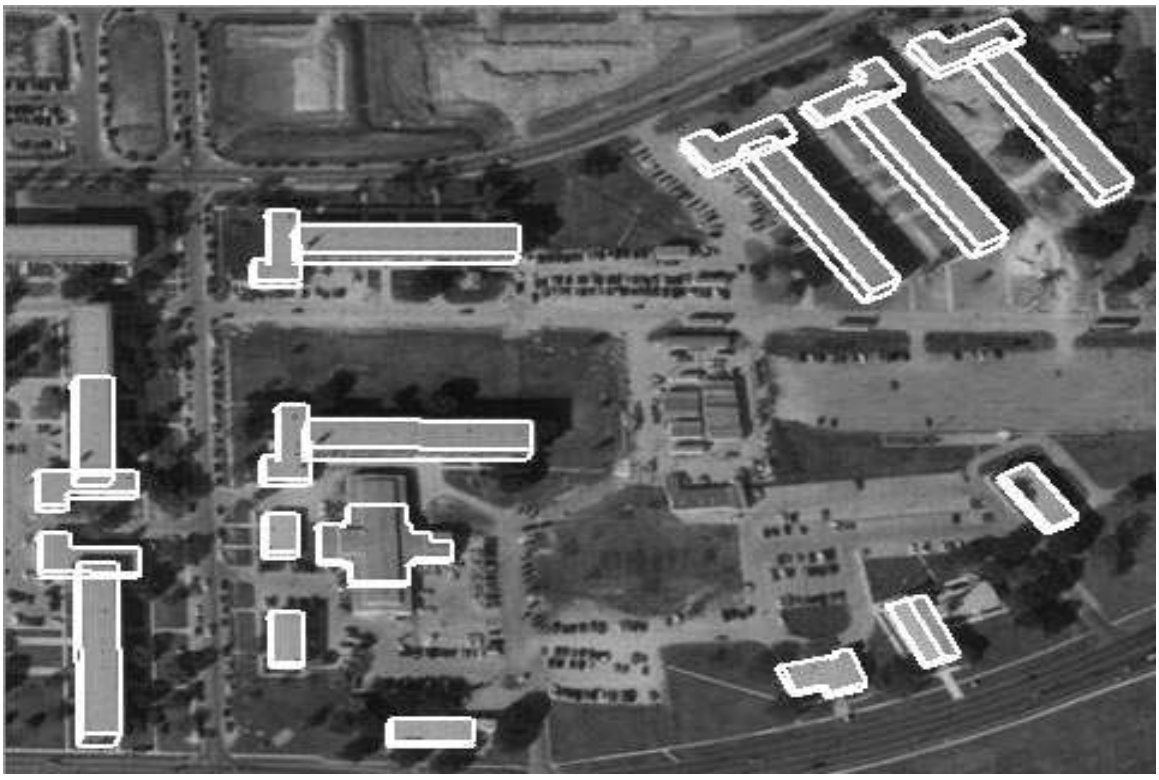

Figure 2.3: An example site model from Huertas and Nevatia [23]. Manually constructed geometry indicated by white lines models most of the 3-d structure of the scene.

Several algorithms take the approach of maintaining a 2-d mosaic of the scene consisting of mixtures of Gaussians and registering new images to the mosaic using a homography to detect changes. Mittal and Huttenlocher [29] register new images to the most probable location on a background mosaic using the Levenberg-Marquardt minimization algorithm. They show qualitatively good results for both a pan/tilt camera setup and a sequence from a hovering aerial platform. Ren et al. [35] developed a similar technique using an estimated background registered with a homography and a mixture of Gaussians approach to change detection. This algorithm is also demonstrated to work with pan/tilt cameras. See figure 2.2 for some some example recovered mosaics from both algorithms. Hayman and Eklundh [19] develop a similar algorithm for pan/tilt cameras for use on a mobile robot observer.

Figure 2.4: A screenshot of the RADIUS site modeling application showing the types of structures that can be manually created including roads, buildings, and water ways (Copyright SRI International 1994).

2.2.2 Manual Site Modeling

These registration techniques only perform well when camera motion is limited to pan/tilt or the scene of interest is mostly planar. When these assumptions can't be met the 3-d nature of the scene has to be taken into account. One technique that was most popular in the 1990's is to do change detection using manually constructed 3-d site models of the scene. In this approach the user constructs a priori a crude 3-d model of the site of interest using specially developed geometry modeling software (see figure 2.3). This 3-d surface is then used to make correspondences between images so that some change detection algorithm can be applied.

One of the first and most developed examples of this approach was the RADIUS project of the early 1990s [12]. Change detection was a major application of this system, designed to aid image analysts in collecting intelligence information from satellite imagery. Several methods for detecting change were tested on this system including comparisons of albedo, edgel texture, edgel orientation, and line orientation. These methods used correspondences from the 3-d site model geometry to compare statistics.

For example in the RADIUS algorithm for comparing albedo, the observed intensity

of an "event" surface E is compared to a "reference" surface R which is known to have the same surface normal and is known to never change. This correspondence of surfaces would be done as part of the construction of the site model. A function of the ratio of the event and reference intensities is computed:

$$log(1.0 + \frac{I_E}{I_R}) \tag{2.2.2}$$

This ratio of intensities is theoretically invariant to global illumination changes so it can be compared to the same calculated ratio from a previously collected image. Significant differences in the computed ratios are marked as a change.

Another relatively recent approach to change detection using site modeling is by Huertas and Nevatia [23], who match detected edges in an image to boundaries of a previously constructed model. Interest in site modeling for change detection has diminished in recent years, with most site modeling literature focusing on automated or semi-manual reconstruction techniques. See the literature review paper by Hu et al. [22] for recent developments in this area. Change detection using manual site modeling never gained popularity among the image analysts for whom it was designed because of the overhead of constructing 3-d geometry. In many ways, the algorithm presented in this thesis is a spiritual successor to the RADIUS work done in the 90's, where instead of a manually constructed 3-d site model an automatically learned 3-d volume reconstruction is used.

2.2.3 Motion Detection

In the case where the imagery is video and the changes of interest are limited to moving objects one can use motion estimates to detect changes. There has been quite a lot of work done on detecting moving objects in aerial video sequences, since for many change detection applications these are the only relevant changes. These techniques can distinguish motion caused by camera movement from motion caused by moving objects on the ground and classify only the later motions as change. However this framework requires high frame rate video to be able to obtain accurate motion estimates. Also many aerial surveillance applications deal only with still imagery and many changes of interest such as new structures being built occur over long periods of time.

Yalcin et al. [44] use optical-flow to register frames of a video sequence to a background mosaic. Then based on the learned background and the flow, the new frame is segmented into background and foreground using an EM-based motion segmentation. They show good results for detecting moving vehicles from aerial video, though it is unclear whether there is significant movement of the aerial platform in their sequences. The fact that they use a background mosaic as part of the foreground segmentation suggests that their method would not perform well in the presence of significant 3-d relief between frames.

Tao et al. [42] present another flow based approach with a tracking application in mind. They use a dynamic layer representation of foreground and background, and update their layer estimates in a MAP framework using motion information. This approach is demonstrated to work well at segmenting moving vehicles in aerial video sequences, however like Yalcin et al. it is unclear whether their technique can handle scenes with significant 3-d relief. Also all motion detection techniques have an intrinsic weakness in being unable to detect objects moving in the same direction as the camera, since they have the same motion vectors as off-planar structures.

2.2.4 Other Techniques

Many other change detection algorithms for aerial scenes have been invented that cannot be put into any of these categories. The difficulty of the 3-d problem has lead researchers to try many new and inventive change detection techniques to sidestep the problem of learning 3-d geometry.

A few algorithms have been developed that extract some kind of scene statistics from each image and look for changes between images. Del Frate et al. [14] classify each pixel in an image as being a certain type of material such as asphalt, building, soil or vegetation using a neural network framework. They then use these classifications to detect large scale changes between images (see figure 2.5). Reno and Booth [36] and Borchani et al. [4] developed similar techniques that classify man-made objects vs. natural objects for change detection. These techniques are demonstrated to work on low-resolution imagery where 3-d relief isn't noticeable enough to cause any problems. Li et al. [28] match line segments between pairs of images and take unmatched lines as changes. The usefulness

Figure 2.5: Change detection results from Del Frate et al. [14]. (Left and Middle) classification maps from 2002 and 2003 respectively where black: asphalted surface, white: buildings, dark gray: bare soil, and light gray: vegetation. (Right) detected changes.

of all of the techniques described above is severely limited by the narrow types of changes they are able to detect.

Heller et al. [20] use stereo pairs of satellite images to reconstruct 3-d geometry of the scene and then compare the reconstructed geometry from different pairs of images to detect 3-d changes to the scene. This idea is a good step forward but its applicability is still limited. Firstly it depends on having stereo satellite pairs and relies on an accurate reconstruction from these images, which may not be realistic. Secondly it cannot detect appearance changes on the surfaces of the scene which may be important sources of information.

2.3 Conclusions

Over the years the Stauffer-Grimson algorithm [41] has emerged as the standard algorithm for doing change detection in 2-d. It employs background modeling which allows a more informed statement about what an abnormal pixel in an image should look like. The mixture of Gaussians distribution used to model pixel behavior can model complicated multi-modal pixel behavior such as water specularities and dense traffic. Most importantly the online updating scheme presented in their paper is fast, accurate, and can automatically adapt to long term changes to the scene.

In the realm of 3-d change detection however, no single technique has stood out as a

comprehensive solution. There are many good techniques that work for pan/tilt cameras but fail when the camera motion is allowed to be fully general. Site modeling algorithms have been developed to cope with the 3-d problem, however the manual labor involved in constructing site models has limited the adoption of these algorithms. Motion based techniques appear to work well for mostly planar scenes, however they are limited to high frame-rate video and may miss moving objects which move in the same direction as the camera relative to the background.

What is needed is a change detection algorithm which performs similarly to the Stauffer-Grimson algorithm but in 3-d. Like the Stauffer-Grimson algorithm it should employ background modeling to learn the normal appearance of the scene from many images, though the concept of "background" is more complicated now that it depends on viewpoint. Such an algorithm should operate under the general requirements outlined in the introduction to ensure that it is usable in a real world system.

Chapter 3

Volumetric Appearance Modeling (VAM)

The current methods for detecting changes in a 3-d environment summarized in the previous chapter all work well under certain conditions, but none are complete solutions to the 3-d change detection problem. To be usable in a real world system, a 3-d change detection algorithm must work under the general conditions outlined in the introduction which include arbitrary viewpoints, arbitrary lighting, and online learning. This chapter presents a new theory for 3-d change detection which satisfies all of these requirements with the exception of variable lighting which will be addressed in chapter 5.

To learn the background appearance of a scene when images can be taken from any viewpoint, some 3-d structural information about the scene must be learned. The most obvious solution is to estimate a 3-d world surface using a stereo or other multi-view reconstruction algorithm. However there are several drawbacks to this approach. The first is the choice of images used in the reconstruction. How does one choose these images and what guarantee is there that the reconstruction will be at all accurate? Another problem is that the world surface may change over time as new structures are built. At what point should the 3-d surface get recomputed to handle these changes and with what images?

This later point suggests that it will be useful to keep hypotheses of multiple surfaces

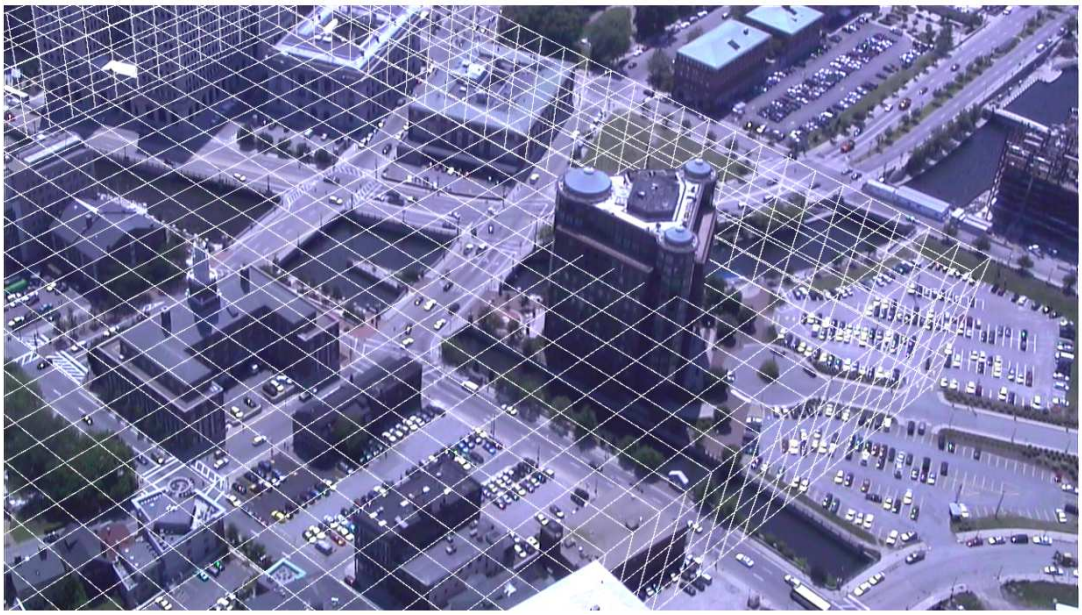

Figure 3.1: An example showing a voxel array drawn on top of a scene. Note that this voxel array does not contain all of the 3-d structure in the scene. The top of the tallest building lies outside the volume and so its background appearance cannot be learned.

around at any time rather than continuing to recompute a single 3-d surface. Taking this idea a step further it would be ideal if one could keep track of hypotheses that surfaces exist at ANY point in the world. At this point it becomes clear that an online solution to the 3-d "background" modeling problem must operate on a volume rather than a surface so that all possible surface hypotheses can be maintained and updated. In addition to the 3-d structure of the world, the appearance of this structure when seen in an image must also be learned.

The framework developed in this thesis will be referred to as **volumetric appearance modeling**, or VAM, as it involves the joint learning of 3-d volume structure and appearance. The scene of interest is modeled as a voxel volume, a representation most widely used for space carving [5] [45] [2]. However where the space carving community is only interested in recovering 3-d structure, in VAM voxels are used to maintain surface and appearance hypotheses at all points in space.

The rest of this chapter is organized as follows. Section 3.1 explains the setup for the full voxel framework. Section 3.2 describes the online updating algorithms for occupancy

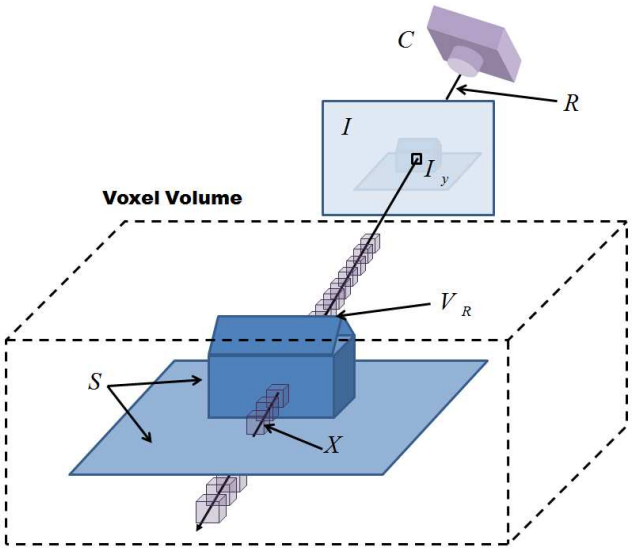

Figure 3.2: Voxel notation. X is voxel on the world surface $\mathcal{S}$ lying on the ray $R(X)$ and projecting into image I at pixel y. However V_R is the voxel that actually produced color I_y.

and appearance probabilities. Equations for detecting changes in images are given in section 3.3 and equations for rendering synthetic images are given in section 3.4. Section 3.5 discusses the theoretical convergence properties of the updating scheme. In section 3.6 compares and contrasts VAM with space carving techniques.

3.1 Framework

To begin, a bounded volume containing all of the 3-d scene of interest is partitioned into voxels (see figure 3.1). At any point in time a voxel X can be either a surface in the world or not surface i.e. empty space or inside a solid object. The state of X being a surface voxel is denoted $X \in \mathcal{S}$. It will prove useful to think of the surface occupancy estimation problem as a Bayesian decision problem, so that it makes sense to talk about probabilities like $P(X \in \mathcal{S})$ which measures one's belief that X is a surface voxel. It will be assumed that the surface occupancies of voxels are independent of each other, a simplification since surface voxels are likely to be near other surface voxels.

Some notation for describing important objects is needed at this point, and a graphical example is given in figure 3.2. For any image I with known camera C the voxel volume

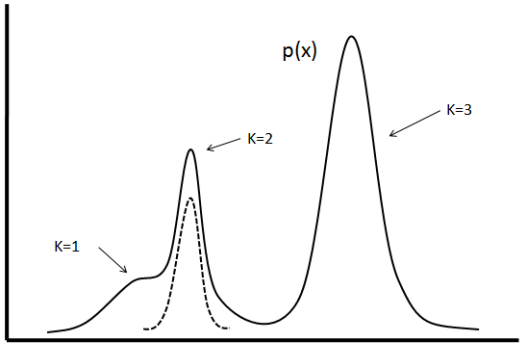

Figure 3.3: A sample mixture of Gaussians distribution with $K = 3$ modes.

can be partitioned into rays of voxels that project into common pixel locations in the image. The ray to which voxel X belongs is denoted by $R(X)$ (or just R when X is not specified) and the color intensity (color for short) to which X maps in the image by I_y where $y = C(X)$, the projection of X in the image. For any ray R the voxels $X' \in R$ can be ordered with the inequality $X'' > X'$, when X'' is further along the ray from the camera center than X'. Also for each ray there may be several true surface voxels along it which project into X, for example in figure 3.2 the ray has three true surface intersections, two on the building and one on the ground. However there is always exactly one voxel which actually produced the intensity seen in the image and this voxel is important enough to have its own name V_R or simply by V when the ray is understood.

In each voxel there will also be stored a mixture of Gaussians distribution to model surface appearance. The mixture of Gaussians is a well-studied and widely used probability distribution for modeling multi-modal data. It has become the standard tool for modeling appearance in stationary camera change detection algorithms such as the classic Stauffer-Grimson algorithm [41] and the improved version used by KaewTraKulPong and Bowden [25].

A mixture distribution [10] is a probability density function of form:

$$p(x|\Theta) = \sum_{k=1}^{K} p(k)p(x|k, \theta_k) \tag{3.1.1}$$

where the $p(x|k, \theta_k)$ are themselves density functions called the "modes" of the mixture

and the $p(k)$ are constants > 0 which sum to 1. The mixture distribution is a mixture of Gaussians when the $p(x|k, \theta_k)$ are independent Gaussian distributions (see figure 3.3). The full distribution with 1-d Gaussian densities in the form that will be used in this thesis is then:

$$p(x) = \sum_{k=1}^{K} \frac{w_k}{W} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}} \tag{3.1.2}$$

where the $\{w_k\}$ are weights such that $W = \sum_k w_k$ and $p(k) = w_k/W$. This weighted form of the mixture differs from standard mixture of Gaussians formulations and will be used to control how much a mode is modified during updating.

A mixture of Gaussians on image intensity will serve as the appearance model in each voxel. This distribution gives the probability that the voxel produced a given pixel intensity, assuming the voxel was actually the surface that it came from. Using the voxel notation this distribution can be written as $P(I_y|V_R = X)$, which is the probability of observing intensity I_y given X is the voxel that produced it. The big "P" in this context is a slight abuse of notation since it is actually a probability density function on image intensity.

The occupancy probabilities in all voxels are initialized with probability $1/nz$, where nz is the number of voxels in the z direction. The appearance model is initially a uniform distribution which assigns a probability of 1 to every pixel intensity, but will be replaced by a mixture of Gaussians after the first observation. Given a sequence of images $\{I_t\}$ with cameras $\{C_t\}$, after observing each image I_{t+1} the surface estimates $P^t(X \in \mathcal{S})$ and color distributions $P^t(I_y|V_R = X)$ for each voxel X must be updated. After these probabilities are updated on I^{t+1} all information from this image has been used and it will never be referred to again. This makes VAM an online algorithm, one of the key requirements outlined in the introduction. The updating procedure is described in the next section.

3.2 Online Updating

Suppose that the voxel model has been trained on t images and now it is time to update the model with image I^{t+1}. It is assumed that a camera model C (a 3x4 camera matrix

usually) for this image is known and accurate to within a pixel. There are many techniques for automatically computing these matrices and a discussion of such is outside the scope of this thesis (see [17]).

What follows next is a presentation of the theory behind updating of voxel appearance and surface probabilities with image I^{t+1}. They key idea is to use current estimates of 3-d structure to reason about occlusion for updating appearance and using current appearance estimates to reason about probable surfaces for updating occupancy probabilities. Actually implementing these update algorithms is another matter entirely, to which chapter 7 is entirely devoted.

3.2.1 Updating the Occupancy Probabilities

When updating the occupancy probability at voxel X, the assumption is made that $I^{t+1}_{C(X)}$ (from here on simply I_X) is the only relevant pixel in I^{t+1} for updating voxel X. With this assumption it is a consequence of Bayes rule that:

$$P^{t+1}(X \in \mathcal{S}) = P^t(X \in \mathcal{S}) \frac{P^t(I_X | X \in \mathcal{S})}{P^t(I_X)} \tag{3.2.1}$$

The cautious reader may be worried about the mix of continuous and discrete random variables in this equation. Rest assured the basic rules for manipulating conditional probability density and mass functions apply when one argument is discrete and the other continuous (one can verify they agree with the measure-theoretic definition of conditional probability, see [43]). After expanding the color probability terms by voxels along the ray $R(X)$ that actually produced the color, the multiplier for $P^t(X \in \mathcal{S})$ in equation 3.2.1 equals:

$$\frac{\sum_{X' \in R(X)} P^t(I_X | V = X') P^t(V = X' | X \in \mathcal{S})}{\sum_{X' \in R(X)} P^t(I_X | V = X') P^t(V = X')} \tag{3.2.2}$$

$P^t(I_X | V = X')$ is computed using the mixture of Gaussians appearance model stored in voxel X' after training on the first t images. The only issue left is how to compute $P^t(V = X')$ and $P^t(V = X' | X \in \mathcal{S})$.

Consider $P^t(V = X')$, which is the probability that voxel X' produced the pixel

intensity in the image considering only the voxel surface probabilities along the ray, and not any intensity information. From a geometric viewpoint, the voxel will produce the pixel intensity if and only if the voxel is on the surface *and* it is unoccluded:

$$P^t(V = X') = P^t(X' \in \mathcal{S})P^t(X' \text{ is not occluded}) \tag{3.2.3}$$

Experiments were done with several occlusion models including those used in [45] [5] and similar results obtained each time, so the model with the most immediate physical interpretation is used here,

$$P^t(X' \text{ is not occluded }) = \prod_{X''<X'} (1 - P^t(X'' \in \mathcal{S})), \tag{3.2.4}$$

which is the probability that all voxels between X' and the camera contain empty space. This completes the definition of $P^t(V = X')$.The equation for $P^t(V = X'|X \in \mathcal{S})$ is the same except that any instances of $P^t(X \in \mathcal{S})$ in the above expression have probability 1. Note that the update equations for $P^{t+1}(X \in \mathcal{S})$ have been derived using only the current surface and color probabilities for voxels lying in the same ray, and so can be computed.

The careful reader will note that the $P^t(V = X')$ as defined will not quite sum to 1, in fact they will sum to:

$$1 - \prod_{X' \in R(X)} (1 - P^t(X' \in \mathcal{S})) \tag{3.2.5}$$

But the $P^t(V = X')$ should sum to one since some voxel along the ray must have produced the observed intensity. Dividing each $P^t(V = X')$ by equation 3.2.5 will ensure that they do sum to one. One can verify that this is probabilistically correct by defining:

$$P^t(V = 0) = \prod_{X' \in R(X)} (1 - P^t(X' \in \mathcal{S}))$$

to be the probability of passing entirely through the world and observing that:

$$P^t(V = X'|V \neq 0) = \frac{P^t(V = X')}{P^t(V \neq 0)} = \frac{P^t(V = X')}{1 - \prod_{X' \in R(X)} (1 - P^t(X' \in \mathcal{S}))} \tag{3.2.6}$$

This correction is a minor detail, and for ease of reading hereafter $P^t(V = X')$ will be understood to really be the normalized probability $P^t(V = X'|V \neq 0)$.

To better understand the update multiplier in equation 3.2.2, the numerator and denominator can be further decomposed. First define the terms:

$$visX^t = \prod_{X''<X} \big(1 - P^t(X'' \in S)\big)$$

$$preX^t = \sum_{X'<X} \left(P^t(I_X|V = X')P^t(X' \in S) \prod_{X''<X'} \big(1 - P^t(X'' \in S)\big) \right) \tag{3.2.7}$$

$$postX^t = \sum_{X'>X} \left(P^t(I_X|V = X')P^t(X' \in S) \prod_{X<X''<X'} \big(1 - P^t(X'' \in S)\big) \right)$$

Here $visX^t$ is the probability of X being unoccluded and $preX^t$ and $postX^t$ are the probabilities of seeing the color when considering only the voxels on the ray above and below X respectively. Then the numerator of equation 3.2.2 can be decomposed as follows:

$$\sum_{X' \in R(X)} P^t(I_X|V{=}X')P^t(V{=}X'|X{\in}\mathcal{S}){=}$$

$$\sum_{X'<X} \left(P^t(I_X|V{=}X')P^t(V{=}X') \right) + \left(P^t(I_X|V{=}X)P^t(V{=}X|X{\in}S) \right) + \sum_{X'>X} \left(P^t(I_X|V{=}X')P^t(V{=}X'|X{\in}\mathcal{S}) \right) =$$

$$\left(preX^t \right) + \left(P^t(I_X|V{=}X){\bullet}1{\bullet} \prod_{X''<X} \big(1{-}P^t(X'')\big) \right) + \sum_{X'>X} \left(P^t(I_X|V{=}X'){\bullet}0 \right) =$$

$$preX^t{+}visX^t{\bullet}P^t(I_X|V{=}X)$$

The the denominator can be decomposed as follows:

$$\sum_{X' \in R(X)} P^t(I_X|V{=}X')P^t(V{=}X') =$$

$$\sum_{X'<X} \left(P^t(I_X|V{=}X')P^t(V{=}X') \right) + \left(P^t(I_X|V{=}X)P^t(V{=}X) \right) + \sum_{X'>X} \left(P^t(I_X|V{=}X')P(V{=}X') \right) =$$

$$\left(preX^t \right) + \left(P^t(I_X|V{=}X)P^t(X) \prod_{X''<X} \left(1{-}P^t(X'')\right) \right) + \sum_{X'>X} \left(P^t(I_X|V{=}X')P^t(X') \prod_{X''<X'} \left(1{-}P^t(X'')\right) \right) =$$

$$preX^t + visX^t P^t(I_X|V{=}X)P^t(X) + visX^t \left(1{-}P^t(X)\right) \sum_{X'>X} \left(P^t(I_X|V{=}X')P^t(X') \prod_{X<X''<X'} \left(1{-}P^t(X'')\right) \right) =$$

$$preX^t + visX^t \left[P^t(X)P^t(I_X|V{=}X) + \left(1{-}P^t(X)\right) postX^t \right]$$

Using these expansions the multiplier in equation 3.2.2 can be rewritten as follows:

$$\frac{preX^t + visX^t * P^t(I_X|V{=}X)}{preX^t + visX^t \left[P^t(X)P^t(I_X|V{=}X) + \left(1{-}P^t(X)\right) postX^t \right]} \tag{3.2.8}$$

It follows that the update multiplier is greater than 1 if and only if X explains the color better than the voxels below it. This expansion will be considered further in section 3.5 on convergence.

3.2.2 Updating the Appearance Models

The updating of the mixture of Gaussians appearance model in each voxel is done using a weighted version of the update scheme in [25]. Under this weighting scheme each new intensity observation used to update the model will come associated with a weight increment dw, to be specified later, which determines how much the new observation will effect the distribution. At all times the current mixture modes are kept sorted by w_k/σ_k, which puts high probability and low variance modes ahead of low probability and high variance modes. To train the distribution on a new observation x^{n+1} with weight increment dw the first mode for which x^{n+1} lies within 2.5 standard deviations of the mean is updated

with the following equations:

$$\begin{aligned} w_k^{n+1} &= w_k^n + dw \\ \mu_k^{n+1} &= \mu_k^n + \frac{dw}{dw + w_k^n}(x^{n+1} - \mu_k^n) \\ (\sigma_k^{n+1})^2 &= (\sigma_k^n)^2 + \frac{dw}{dw + w_k^n}((x^{n+1} - \mu_k^n)^2 - (\sigma_k^n)^2) \end{aligned} \tag{3.2.9}$$

These equations are weighted versions of the update equations given in [25], which will be the same in the case that the weights are constant. If x^{n+1} is not within 2.5 standard deviations of any mode, the least probable mode is destroyed and replaced with a new high variance mode with mean x^{n+1} and weight dw. Note that with these update equations the w_k can become arbitrarily large however they are normalized before they are used in the mixture distribution in equation 3.1.2. For very large image sets it may be a good idea to bound the w_k, otherwise the distribution will become locked into a particular state and it will require many many images to adapt to new changes.

Putting this back in the voxel framework, the appearance model in each voxel X is updated with the pixel intensity I_X seen at its projection in the image, using equation 3.2.9. But X may or may not be the voxel that actually produced the observed pixel intensity, and it would be undesirable to train a surface voxel on this intensity if it is in fact occluded in this view. The weighted modification to the update equations was introduced to handle this problem of many voxels projecting into the same pixel.

Consider a ray R containing voxels $\{X_i\}$. It is not known for certain which voxel produced the color in the image, but the probability of each voxel X_i producing the color, $P_t(V = X_i)$, can be computed. To be fair, all of the voxels $\{X_i\}$ along the ray must be trained on the color seen, but how little or how much a color effects a voxels Gaussian mixture model can be controlled by choosing appropriate update weight increments, and the natural answer is to use the $dw = P_t(V = X_i)$ as such weights, so that a voxels color model is updated proportionately to how likely it is that the voxel produced the color seen. For a sometimes occluded surface voxel this scheme prevents occluding surface colors from seriously corrupting the voxels color model. For example the voxel X in figure 3.2 is occluded from the shown viewing direction, so it is undesirable to train X on the

occluding roof's color. Training with the $P_t(V = X_i)$ as weights guarantees the influence of the roof's color on X's appearance model is minimal.

3.3 Detecting Change

Suppose that the model has been trained on an image sequence and it is desired to detect change in a new image I with camera C. Consider a pixel I_X which backprojects into a ray R of voxels $\{X'\}$. The probability of seeing this color is then,

$$P^t(I_X) = \sum_{X' \in R} P^t(I_X|V = X')P^t(V = X') \tag{3.3.1}$$

which are all computable quantities. These probabilities are computed for each pixel in the image and every pixel with probability less than some fixed threshold ϵ is considered change. The output is a binary mask for the image representing the detected change. Pixels with high probability relative to the learned voxel model will be marked as background while those with low probability will be considered foreground and therefore change.

Alternatively, some authors on background modeling like to formulate the change detection problem as a Bayesian decision problem [10]. In this case one wishes to assign each pixel a label B for background or F for foreground which have prior probabilities $P(B)$ and $1 - P(B)$ respectively. Given a pixel observation O the probability that it is background is:

$$P(B|O) = \frac{P(O|B)P(B)}{P(O|B)P(B) + P(O|F)(1 - P(B))} \tag{3.3.2}$$

In this equation $P(B)$ can be considered to be a fixed constant and the distribution $P(O|F)$ of foreground intensities can be assumed to be uniformly distributed over the color range [0,1]. In the context of VAM the probability $P(O|B)$ is the same as the probability $P^t(I_X)$ which earlier in this section was being thresholded to label changes. It turns out that thresholding $P^t(I_X)$ and thresholding $P(B|O)$ in this Bayesian decision framework are equivalent, albeit for different thresholds. This is because equation 3.3.2 for $P(B|O)$ is a monotone increasing function of $P(O|B)$ when $P(B)$ is fixed and $P(O|F)$ is uniform. If

however one can say more about $P(O|F)$ than it being a uniform distribution or if $P(B)$ is known to vary at different points in the scene then the Bayesian approach can produce more accurate results.

3.4 Rendering Synthetic Images

Though VAM was designed for detecting changes in a new image, the background 3-d structure and appearance learned from images can also be used to render synthetic images from unseen views. The process for doing this is almost identical to detecting changes. However instead of summing the appearance probabilities along a ray, one computes the expected appearance along the ray:

$$E^t(I_X) = \sum_{X' \in R} E^t(I_X|V = X')P^t(V = X') \tag{3.4.1}$$

Some examples of synthetic renderings are given in the next chapter. These renderings are useful for visualizing what appearance has been learned after training on some number of images.

3.5 Convergence

A key question is under what conditions this simultaneous modeling of geometry and appearance is guaranteed to converge, which shall be defined to occur when empty space voxels have their surface probabilities go to 0 and true surface voxels have their surface probabilities go to 1 and color models go to a single Gaussian mode with the minimum allowed variance (a parameter set to ensure that the variance of a mode doesn't become infinitely small). The convergence of a voxel is highly dependent on the range of viewing directions and local color variation in the world and in general cannot be met for every voxel in a scene. For example the exact surface of a uniformly colored roof viewed only from the air cannot be determined, as many voxels lying above and below the actual roof have color models identical to those of the true surface voxels. Still, general conditions can be derived that guarantee a voxel's convergence. The case of convergence of a surface

voxel X is discussed in detail below, and the argument for an empty space voxel is similar.

The first assumption made is that when it is unoccluded the surface at X projects the same color in each image, though in practice convergence can still occur when there are minimal variations in observed color. Suppose first that X is unoccluded in all views. Then X will see the same color in each image and eventually $P^t(I_X|V = X)$ is guaranteed to be maximal (as the mixture model variance is capped in practice). In particular it will be greater than $postX^t$, so the update multiplier in equation 3.2.8 will be greater than 1 and the surface probability for this voxel will be increasing with each new image. Increasing, but not necessarily to 1, and in regions of ambiguous projection in the world it will converge to something less than 1. The denominator of equation 3.2.8 can be rewritten as:

$$\begin{aligned} preX^t + visX^t * P^t(I_X|V = X) + ... \\ visX^t\big(1 - P^t(X)\big)\Big[postX^t - P^t(I_X|V = X)\Big] \end{aligned} \tag{3.5.1}$$

and after substituting into equation 3.2.8 it is clear that the surface probabilities for X as a function of images trained on are a sequence of form:

$$p_{t+1} = p_t \frac{a_t}{a_t - b_t(1 - p_t)} \tag{3.5.2}$$

where:

$$\begin{aligned} a_t &= preX^t + visX^t * P^t(I_X|V = X) \\ b_t &= visX^t\Big[P^t(I_X|V = X) - postX^t\Big] \end{aligned} \tag{3.5.3}$$

This sequence converges up to 1 provided that the $\{a_t\}$ are bounded from above (automatically true) and the $\{b_t\}$ are bounded from below by an $\epsilon > 0$, i.e. the voxel X consistently explains observations better than the voxels below it. To prove the convergence given these conditions, it is enough to show that for any $0 < q < 1$ there is a T such that $p_t > q$ for all $t > T$. Suppose for some t that $p_t < q$, then:

$$p_{t+1} = p_t \frac{a_t}{a_t - b_t(1-p_t)} > p_t \frac{a_t}{a_t - \epsilon(1-q)} > p_t c \tag{3.5.4}$$

for some constant $c > 1$. It follows that:

$$p_{t+s} > p_t c^s \tag{3.5.5}$$

which will eventually be $> q$ for large enough s.

So there are two conditions for convergence of the voxel's surface probability to 1. Firstly $visX^t$ must be bounded from below, which will be automatically true for a voxel unoccluded in all views as the empty space voxels above it have decreasing surface probabilities via a modification of the above argument. Secondly $P^t(I_X|V = X) - postX^t$ must be bounded, requiring the voxels lying below the surface at X to be trained on some range of colors distinct from the color at X. This means that the world surface in the neighborhood of X must have some amount of color variation AND that cameras taking the images must be spread out enough to project this variation into voxels around X. If these conditions are met, X will converge.

If X is occluded in some views, by what has been shown above the occluding voxels will converge after enough images, preventing the occluding surfaces' colors from corrupting the color model at X. After sufficiently many images of the true surface color, the mixture distribution at X will be maximal, and the condition is the same as before, and the surface probabilities here will again converge to 1. And so the following has been proven:

Convergence Theorem 1. *A voxel X lying on a world surface will converge provided that the surface images a constant color in all unoccluded views, and that all voxels X' near X lying beneath the surface frequently project into some pixels of sufficiently different color from the true surface color at X.*

To conclude this section it is worth mentioning a few cases where true surface voxels will not converge because the assumptions of the theorem are violated. Many surfaces, such as the insides of buildings, will always be occluded in images and so their surface probabilities will not converge to 1. These are legitimate surfaces, but their failure to converge is not a problem since they will never be needed in a change detection calculation because they are

never seen. If a surface is in a uniformly colored region such as a road then its occupancy probability will not converge to 1. More will be said about this case in the next chapter. Finally the converge theorem only applies for an infinite sets of images. In real finite image sequences, rarely seen surfaces like narrow alleyways between tall buildings may not be observed frequently enough to ever converge.

3.6 Comparison with Space Carving

As the VAM framework uses voxels to model 3-d geometry it is worthwhile to compare and contrast it with space carving algorithms. These algorithms have been around for about 10 years and they use a voxel volume as a framework for doing a 3-d reconstruction. Early voxel-based reconstruction algorithms include work done by Seitz and Dyer [39] and Culbertson et al. [7], however the space carving framework was first formalized in 2000 by Kutulakos and Seitz [27]. This algorithm iteratively projects each voxel into all images and determines whether the set of pixel intensities observed are photo-consistent. If not then the voxel is removed from the volume. One of the significant shortcomings of this algorithm is that if a voxel is removed by mistake then voxels behind it may also be removed, resulting in a hole being punched all the way through the model.

Several probabilistic reworkings of the space carving were done in the following years, largely to address this hard voxel/no-voxel assignment that results in holes. Broadhurst et al. [5] present a probabilistic algorithm that learns occupancy probabilities at each voxel and from these constructs a maximum likelihood surface. They also show how to render synthetic images from this volume, much like is done in VAM, however for some reason they don't use an occlusion model in their rendering, meaning it will not perform well on scenes with significant occlusion. Some similar probabilistic space carving papers include Agrawal and Davis [2] and Yao and Calway [45].

Despite both having a voxel framework, VAM has as many differences as similarities with these space carving algorithms. Where the original space carving algorithm starts with an occupied world and removes inconsistent voxels, VAM instead starts with a mostly empty world and increases the probability of consistent voxels. VAM is more similar to the

probabilistic space carving formulations, since both maintain occupancy estimates at each voxel and have probabilistic occlusion models. However the key differences are that VAM is *online* and has a *more sophisticated appearance model* which can both filter outliers and learn multi-modal appearances. These are necessary requirements for a change detection algorithm that are not needed in a pure 3-d reconstruction algorithm for a fixed set of images.

A natural question to ask is how accurate the geometry recovered by VAM is compared to that recovered by a space carving algorithm. The space carving algorithm would likely have an advantage since it can refer to all the images at once rather than sequentially like VAM. However the mixture of Gaussians appearance model in VAM is more robust to outliers so VAM may produce more accurate reconstructions in non-static scenes containing moving cars or pedestrians. Ultimately the accuracy of the recovered geometry is unimportant to VAM so long as the change detection results produced are correct.

3.7 Summary

In this chapter a novel framework for 3-d change detection called volumetric appearance modeling (VAM) has been introduced. It uses a voxel model to represent both 3-d structure and appearance, with each voxel containing both an occupancy probability and a mixture of Gaussians appearance model. Learning the "background" with this framework is a joint estimation problem of simultaneously learning the 3-d surface and its appearance. An online algorithm for this estimation has been presented and its convergence properties studied. Finally once the model has been trained on enough images, detecting changes in a new image is as easy as marking those pixels which have low probability with respect to the model.

The performance of this algorithm on real data will be explored in the next section, but it is already clear that this technique has many advantages over previous approaches. In particular it meets all of the usability requirements outlined in the introduction. It is fully automatic requiring no manual construction of 3-d geometry, it can use images taken from arbitrary camera positions, and it works online so that the model can be updated

with new images as they are taken. As such VAM has the potential to become a standard technique for a wide range of real world change detection applications.

Chapter 4

Experiments

In this section, the VAM framework is used detect changes in several synthetic and real world image sequences. These scenes cover a wide range of urban environments which contain vegetation and architecturally complex buildings. In addition the images were often taken from low viewing angles, up to 45° off nadir. Standard change detection algorithms will not function under such conditions yet the VAM framework performs very well.

This chapter is organized in two major sections. In the first section the VAM framework is used to detect changes in several example image sequences and the overall results analyzed qualitatively and quantitatively. In the second section, the main sources of errors are investigated in more detail: image misalignment, non-Lambertian surfaces, and sampling deficiency.

4.1 Example Sequences

Three sets of images are used in the experiments in this section. The first sequence is of a realistic miniature town made from model railroad kits. The next three sequences are aerial images taken from helicopter at different locations in Providence, RI. In all of these sequences, the images were taken at roughly the same time so that illumination is constant. Camera matrices for all images were computed from manual correspondences to within 1 pixel reprojection error using standard techniques [17].

In each experiment, a voxel volume is trained on a subset of the images using the VAM updating algorithm from chapter 3. Changes are detected in 5-6 new unseen images of the scene for which ground truth has been marked by hand. Changes are detected at different thresholds and the error rates at different thresholds averaged over all the new images are used to draw out a receiver operating characteristic (ROC) curve. These curves plot the fraction of correctly detected changes (relative to total pixel area) against the fraction of falsely detected changes. The false positive error rate is the fraction of mislabeled pixels to the total number of pixels. The ROC curve is a useful graph for analyzing the trade off between having more correct detections and having fewer false detections.

In many experiments, results are also given for a world model consisting of only a single plane of voxels hand-aligned with the ground plane of the scene. In this special case there is no 3-d geometry and VAM is functionally equivalent to a mosaic based change detection algorithm such as [29] or [35] except that the known cameras and ground plane (which together define a homography) are used to align images rather than an image-correspondence based technique. This planar algorithm makes fewer false detections than one might expect in the presence of significant 3-d relief because most scenes are composed of large regions of uniform color. To see why a planar algorithm can perform well in such a situation consider a simple scene consisting of a pure black road surrounding a pure white building. A black voxel on the road is sometimes occluded by the white building, but since this occlusion happens often enough the mixture of Gaussians in the road voxel can learn white as part of the normal appearance. Since white is normal appearance it will not be marked as change in images where the building occludes the road. However a white car driving along the black road will also be considered normal and will not be marked as change. In general the planar algorithm may be able to learn the appearance of multiple surfaces in each voxel but it loses the ability to detect true changes which appear similar to these occluding surfaces.

4.1.1 Plasticville

The first sequence is a collection of 46 images taken of a lifelike miniature town constructed from detailed "plasticville" model railroad kits and imaged under a diffuse lighting condi-

tion (see figure 4.2). Model railroad kits are ideal for a testing scene because detail and realism are what railroad enthusiasts look for in their models. Buildings were painted with flat paints to guarantee that all surfaces are approximately lambertian. Each image was taken with a different arrangement of miniature cars and other clutter to simulate a dynamic environment. This scene is a close substitute to real aerial imagery and is convenient for conducting experiments in a controlled laboratory setting.

In the first experiment VAM was trained on 40 images and then changes detected in the remaining 6 images. A volume rendering of the recovered occupancy probabilities after training is shown in figure 4.1. Note that there is low probability in regions of uniform texture such as roads and roofs because in these areas there is a layer of voxels which together have high probability (see figure 4.3). Some sample images, expected images from those viewpoints, and detected changes are shown figure 4.2 which demonstrates the high accuracy of VAM. The true changes in the scene are correctly marked and all false detections occur along sharp edges in the images. These false detections are not entirely unexpected as similar "misregistration" errors occur frequently in change detection algorithms for pan/tilt cameras [11]. These false detections occur because the coarseness of the voxel grid and small errors in camera calibration can cause voxels along these edges to be trained on a wide range of colors, giving them low occupancy probability and a poor appearance model. More will be said about these misregistration errors later in this chapter.

A voxel world consisting of just a single plane was also trained and tested on these images for comparison. Sample results from the two algorithms are shown in figure 4.4. VAM produces far fewer false detections and visually the true detections appear more crisp and complete. Note that the planar result misses the top middle dinosaur entirely because the mixture models at that location have been trained on the similarly colored (in grayscale) trees and buildings.

In the second experiment, both VAM and planar change detection algorithms are trained on 40 images and changes detected on three different 5-image sets taken at varying viewing angles. ROC curves are given in figure 4.5. VAM performs far better at all angles and though the accuracy of both algorithms degrades as the cameras move from a high

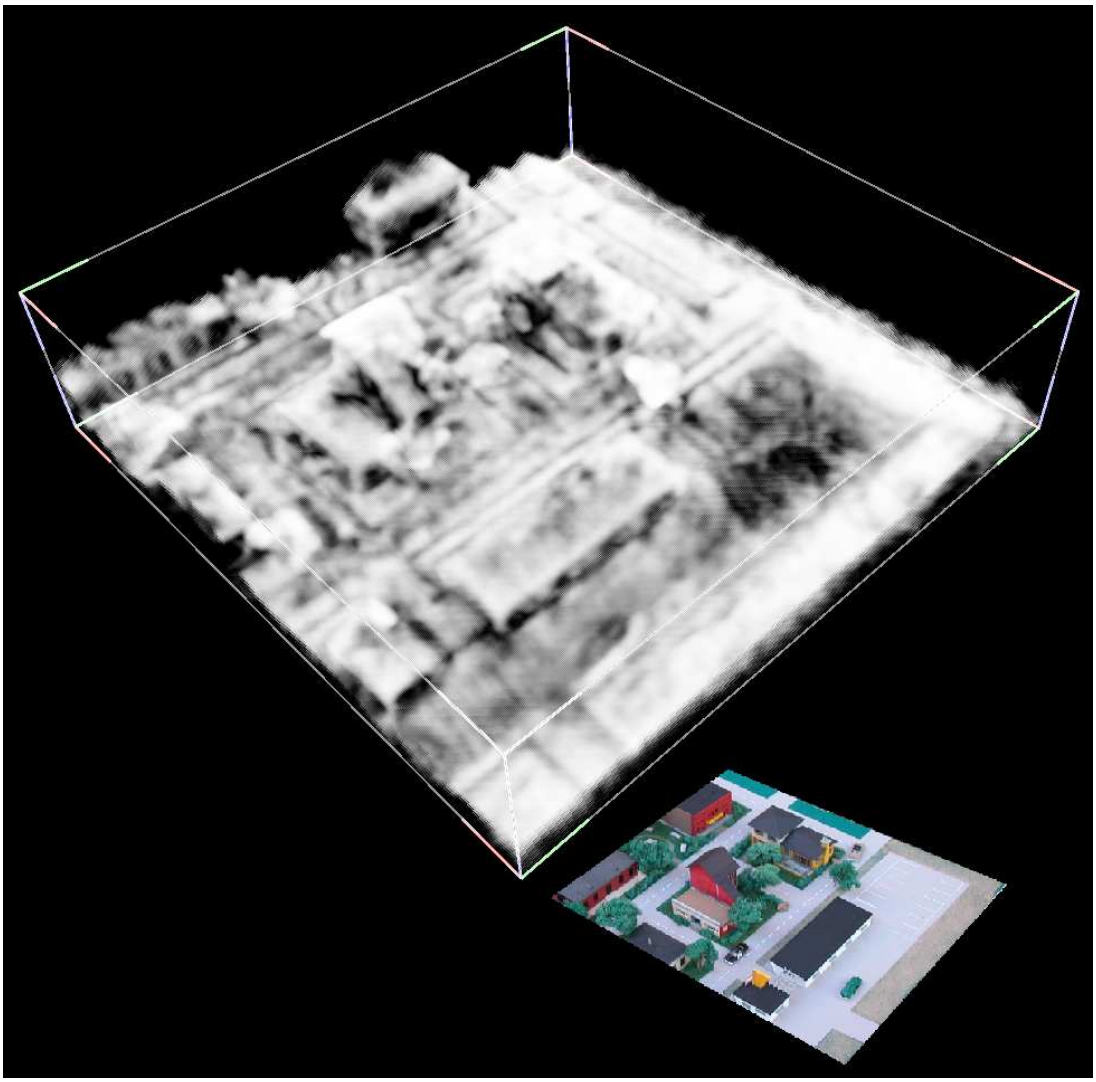

Figure 4.1: A volume rendering of the recovered voxel geometry from the plasticville scene after training on 40 images. White voxels have higher probability. Though this still image doesn't do it justice, the recovered geometry is close to the true 3-d surface except in areas of uniform texture like the parking lot where there is a layer of low probability voxels which together have high probability.

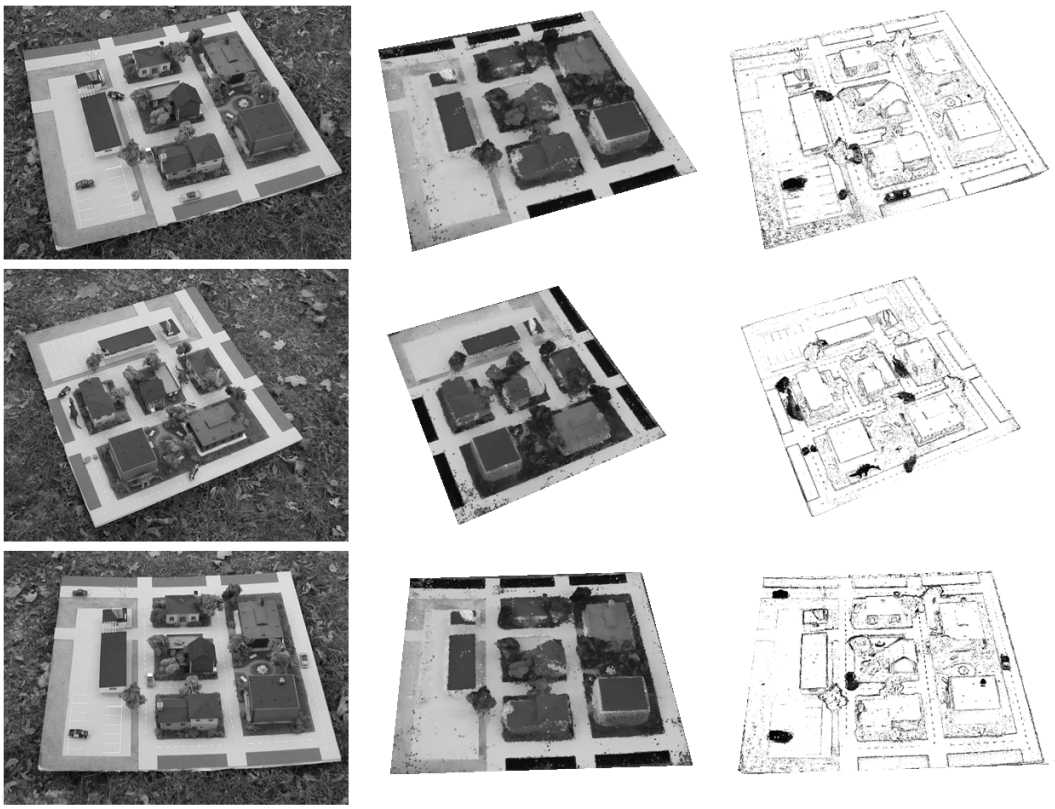

Figure 4.2: Examples from VAM trained on the plasticville model town sequence. (left column) a new unseen image, (middle column) the expected appearance from this viewpoint, (right column) the changes detected with VAM.

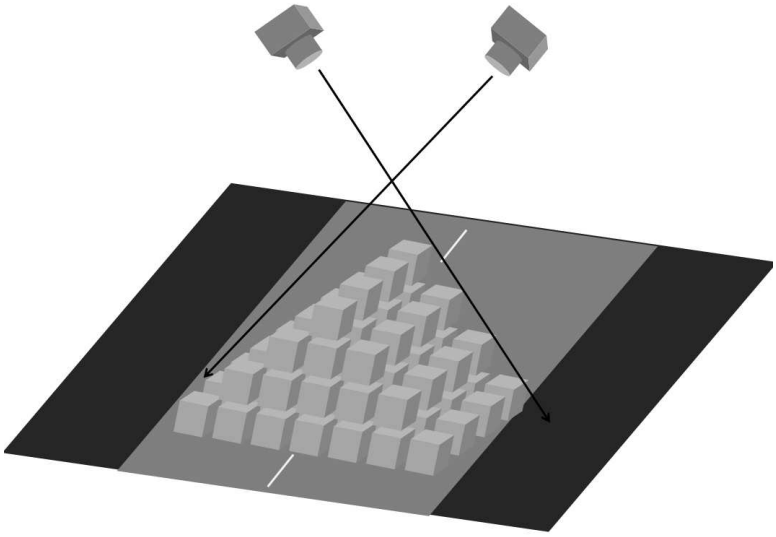

Figure 4.3: For uniformly colored surfaces such as roads imaged from a limited range of viewpoints the shape of the true surface is ambiguous. For the camera angles in this example a surface above or below the actual road could all have produced the observed images. When VAM is run on the images, the recovered voxel geometry will be a layer of voxels with low probability which together add up to high probability.

angle to a more grazing angle, the planar algorithm performance degrades more rapidly.

4.1.2 Steeple Street

The VAM framework was also tested on real aerial imagery taken from helicopter in downtown Providence, RI. The first such set of images is of an office building on Steeple St., which has an intricate steeple on its roof and is bordered on all four sides by busy streets. The only changes in this sequence are the cars driving by on the street. 39 images were taken of the scene from viewpoints in a circle around the building.

VAM was trained on 34 of the images and changes were detected in the remaining 5. The voxel world used was 100 voxels high and 1000 voxels wide on both sides. Some sample images, expected images from those viewpoints, and detected changes are shown figures 4.6 and 4.7. The results are similar to the plasticville imagery, with most vehicles detected correctly and sparse false detections along sharp edges. However, two additional sources of error arise in this real world scene. The first is that some vehicles are not fully detected because their appearance in grayscale is very near to the color of the roads. These

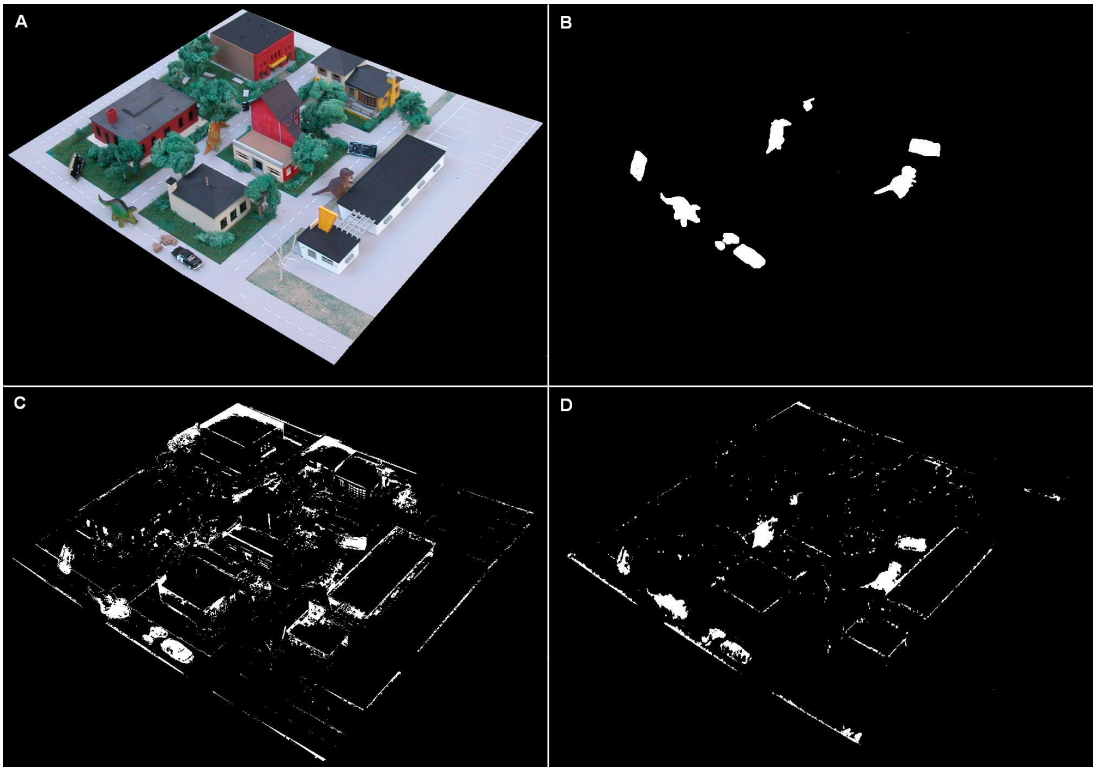

Figure 4.4: Change detection results for VAM and a planar algorithm applied to the plasticville image sequence. A) an unseen image, B) hand-marked ground truth changes on this image, C) results from a planar change detection algorithm, D) VAM change detection results. The planar algorithm has significant errors do to unmodeled geometry, while VAM produces scattered small misregistration errors.

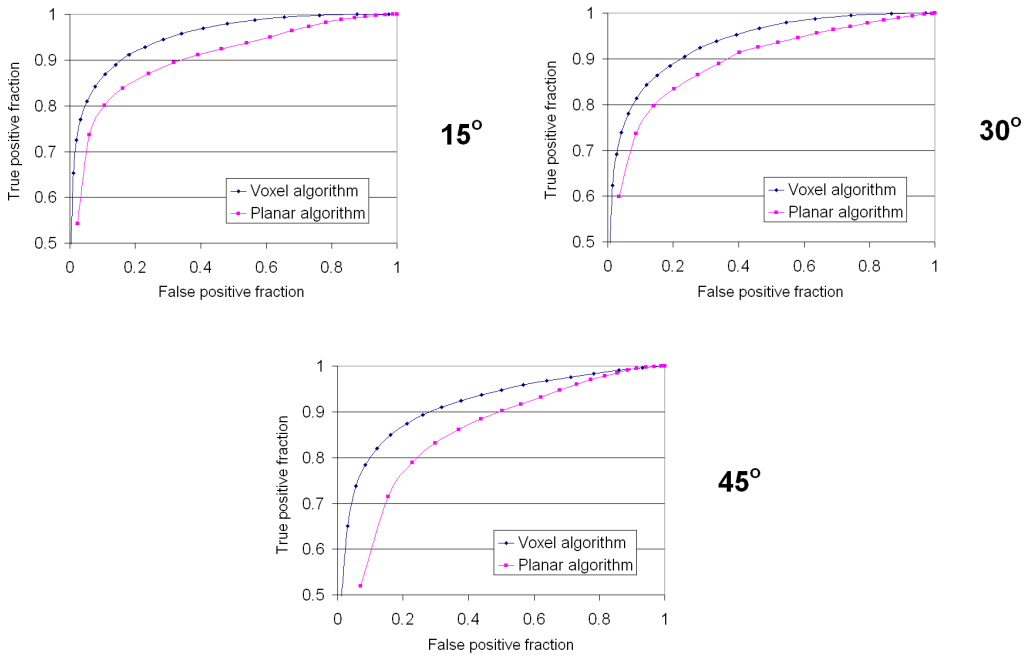

Figure 4.5: Change detection ROC curves for the model town sequence. Results are given for different viewing angles, measured from the normal to the ground plane. Voxel algorithm gives superior all around performance, with a greater difference at larger angles.

missed detections are unavoidable in grayscale imagery but should be detectable with an RGB version of the VAM framework. The second source of error is in non-Lambertian surfaces which can have very different appearances when viewed from different directions. Such errors occur on the metal roof of the office building because the mixture of Gaussians appearance model cannot entirely account for the range of observed intensities. Errors caused by non-lambertian surfaces will be analyzed more later in this chapter.

The planar world was also trained on these images and the results compared to VAM. The voxel algorithm outperforms the planar algorithm by a wider margin in this real world scene as demonstrated by the ROC curves for both sequences shown in figure 4.8.

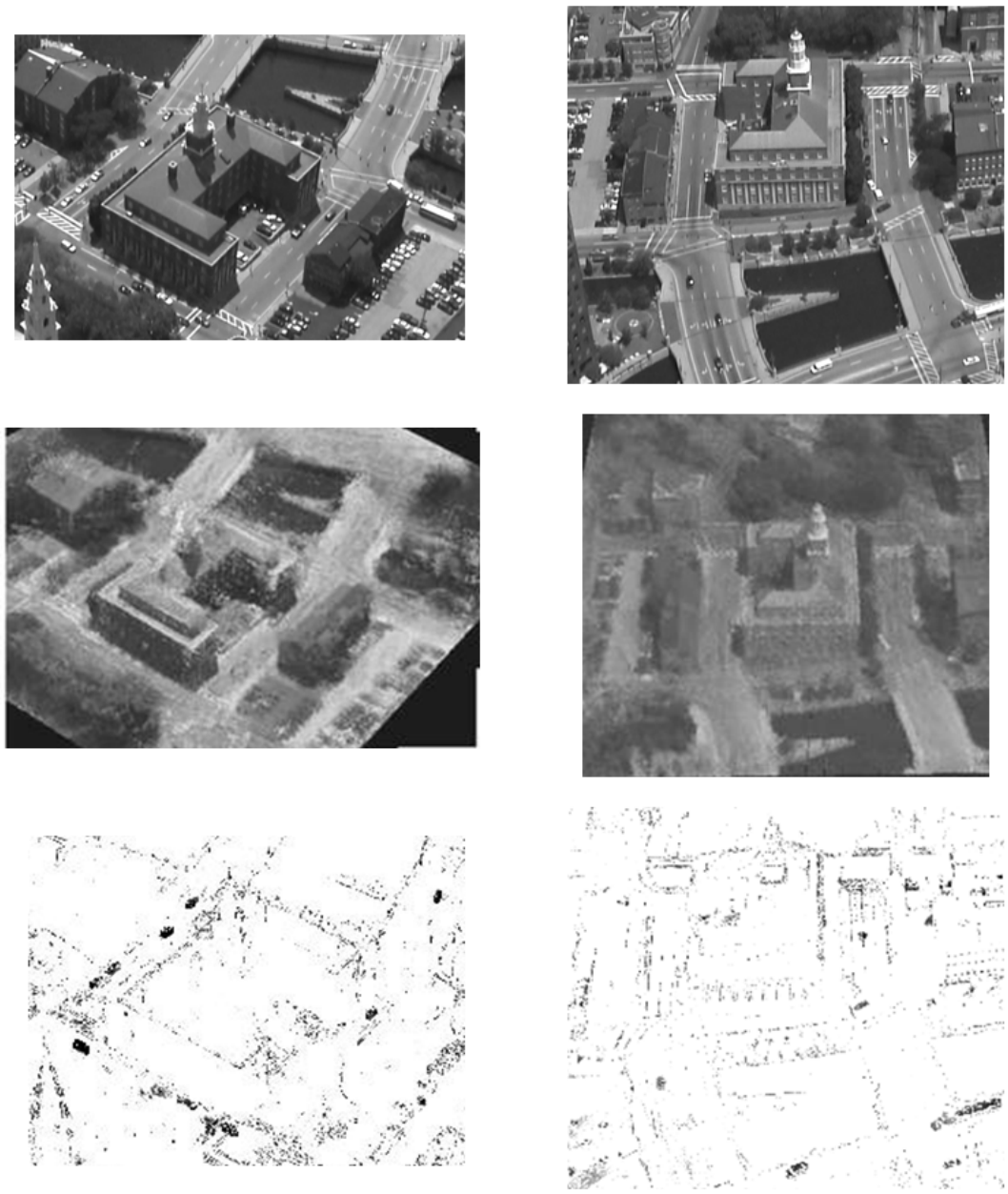

Figure 4.6: Examples from VAM trained on the Steeple St. sequence. (top) a new unseen image, (middle) the expected appearance from this viewpoint, (bottom) the changes detected with VAM.

Figure 4.7: An example of the difficulties VAM has with non-Lambertian surfaces, where false detections are made on the metal roof from a certain viewing direction. (top) a new unseen image, (middle) the expected appearance from this viewpoint, (bottom) the changes detected with VAM.

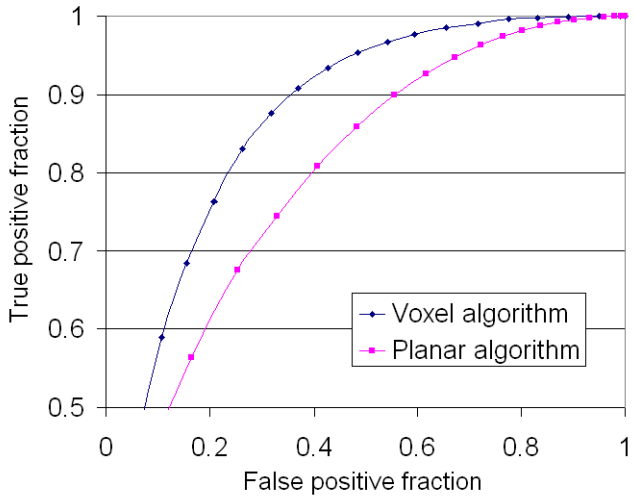

Figure 4.8: ROC curves comparing the performance of VAM vs the planar change detection algorithm on the Steeple St. sequence. Violations of the constant color assumption due to non-lambertian surfaces cause more false positives in this sequence than for the plasticville sequence, but the voxel algorithm still outperforms the planar algorithm significantly.

4.1.3 Post Office

The next image sequence is of a post office building in Providence which was also taken by helicopter. Sample results are given in figure 4.9 and an ROC curve comparing VAM to the planar algorithm is shown in figure 4.10. Again, the vehicles driving around the nearby streets are detected correctly, while misdetections are limited to sharp edges and rare specularities.

4.1.4 Capitol Building

The final helicopter sequence is of the Rhode Island state capital building. This sequence was shot from a relatively low view angle and offers extreme 3-d relief over the course of the sequence. Both VAM and planar algorithms were trained on 30 images from a video sequence and then afterward tested on 3 new images from different points in the video. Sample results comparing VAM to the planar algorithm in this sequence are given in figure 4.11 and an ROC curve comparing VAM to the planar algorithm is shown in figure 4.12. The planar algorithm does not fail completely here because there is not a huge range of viewpoints, but the results are far from usable. VAM has no problem with the extreme

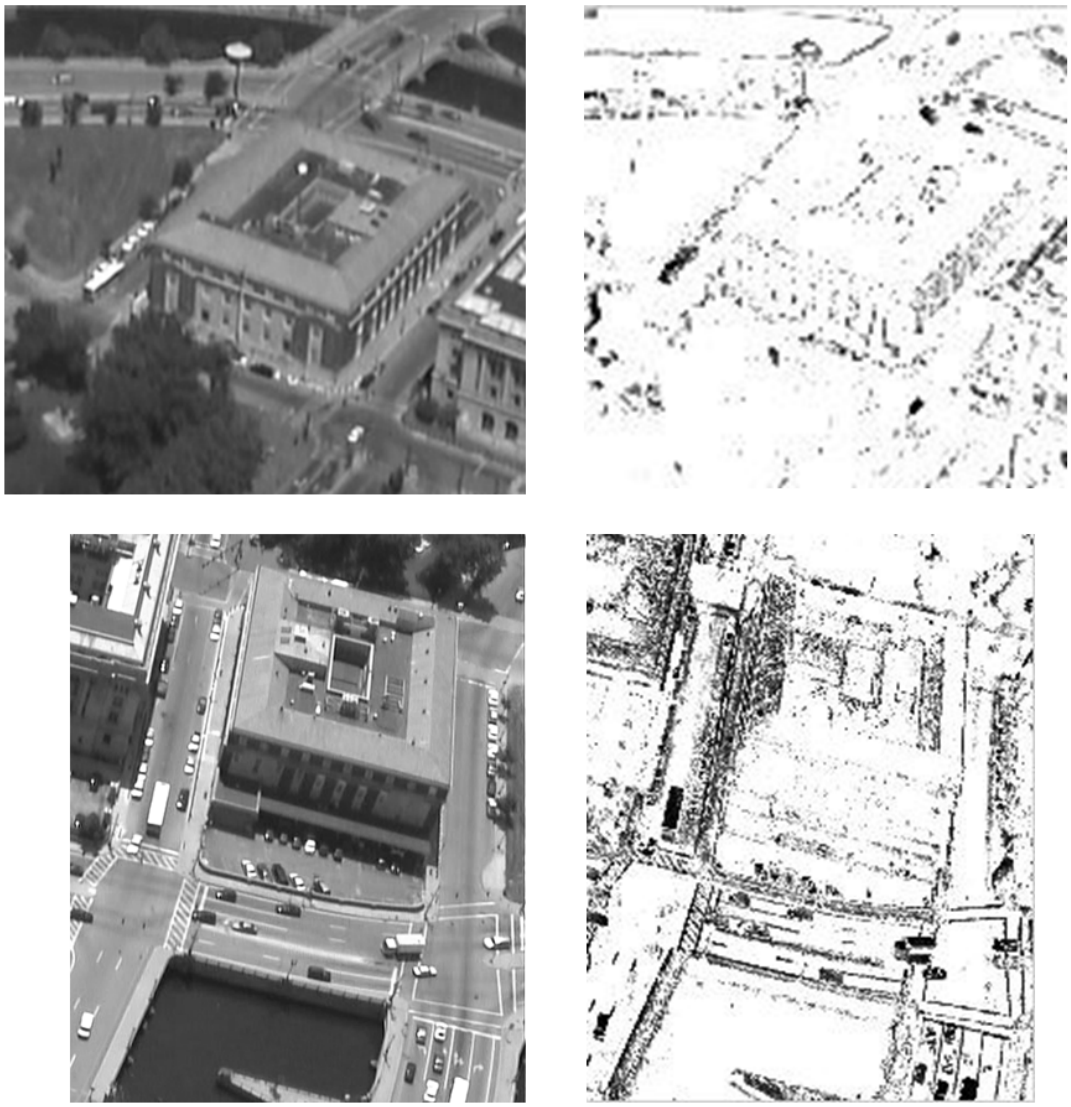

Figure 4.9: Two examples from VAM trained on the post office sequence. Cars driving around the building are correctly detected as change in both examples. The specularities on the roof that occur in the right image cause additional false detections.

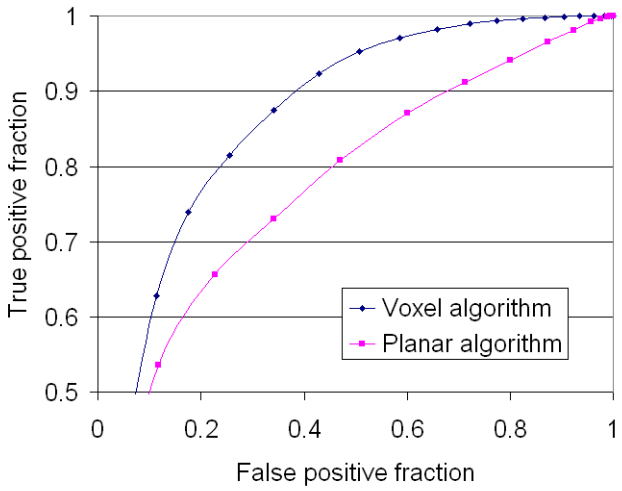

Figure 4.10: ROC curves comparing the performance of VAM vs the planar change detection algorithm on the post office sequence.

3-d relief and produces qualitatively and quantitatively good results. A volume rendering of the recovered 3-d scene in figure 4.13 shows the accuracy of which VAM learns complex 3-d geometry.

4.2 Analysis of Errors

The results of these test sequences show a dramatic improvement in accuracy of the VAM framework over a widely-used planar change detection algorithm. In particular for reasonable thresholds there are no false detections due to 3-d relief caused by different viewpoints. This is a huge step forward since the performance of previous change detection techniques has been limited by the amount of camera motion allowed.

There are a few sources of error which prevent the change detection results from being perfect. There are frequent errors along sharp edges in the images caused by misalignment of pixels and voxels. There are sporadic false detections of specularities on surfaces such as metal roofs and car windows. Finally for shorter image sequences the lack of samples may cause false detections on rarely seen surfaces, because they are not seen often enough to be learned. This section analyses these types of errors in greater detail.

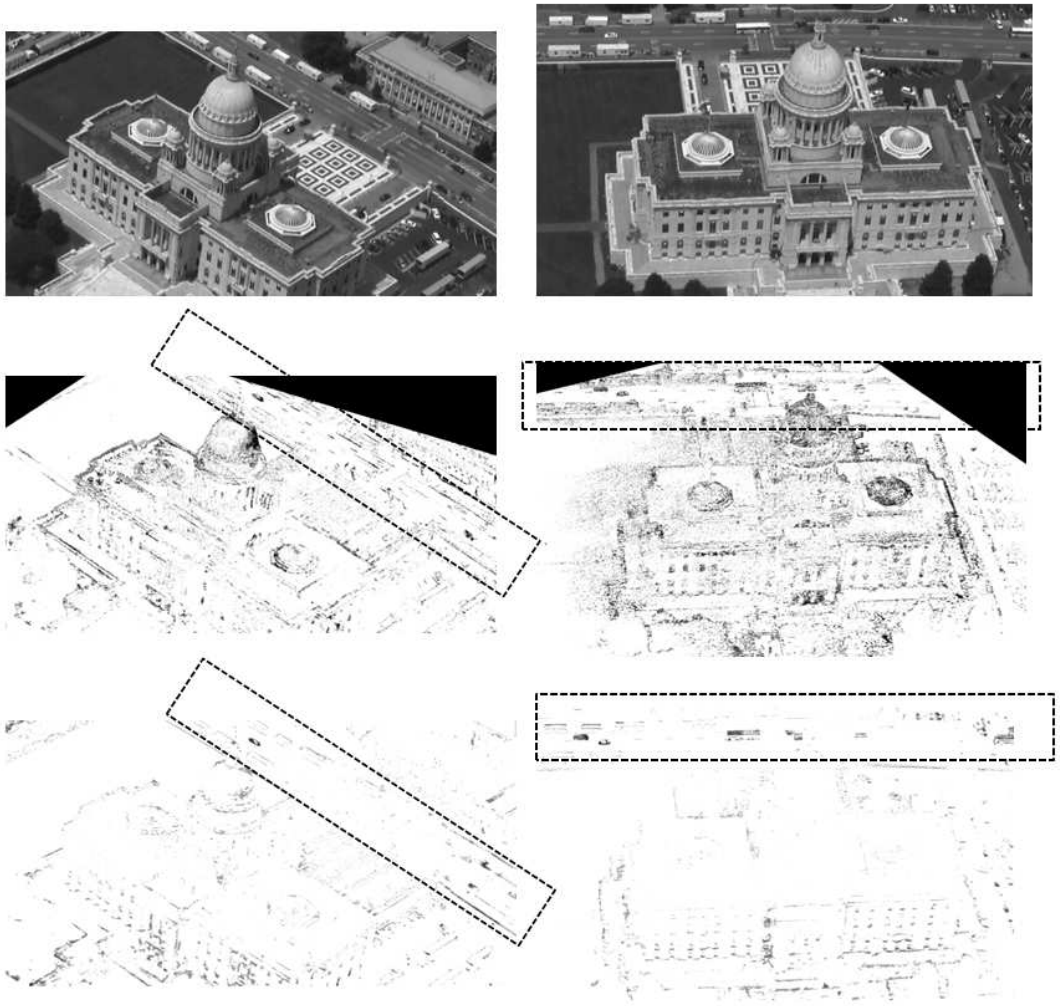

Figure 4.11: Examples from VAM and the planar algorithm trained on the capital building sequence. (top) a new unseen image, (middle) the detected changes using the planar algorithm, (bottom) the changes detected with VAM. The only true changes in this scene are the vehicles passing in the far street (inside dashed boxes) which are easily detected by VAM without many false alarms on the capitol. The planar algorithm fails to cope with the extreme viewpoint shift over the sequence.

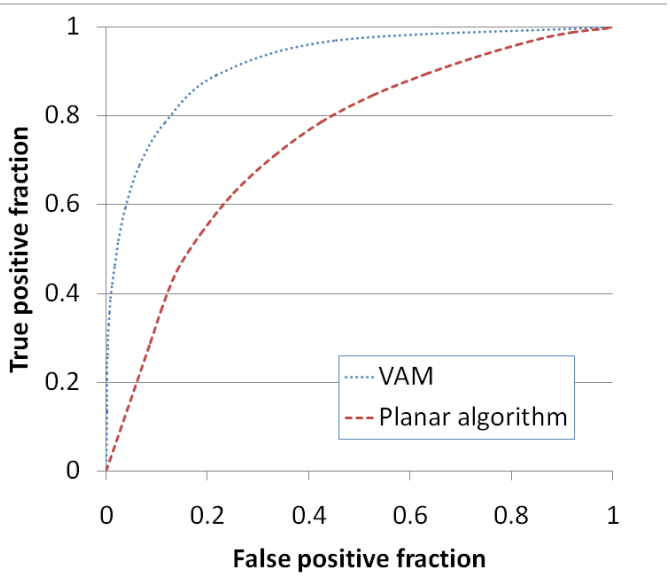

Figure 4.12: ROC curves comparing the performance of VAM vs the planar change detection algorithm on the capitol building sequence.

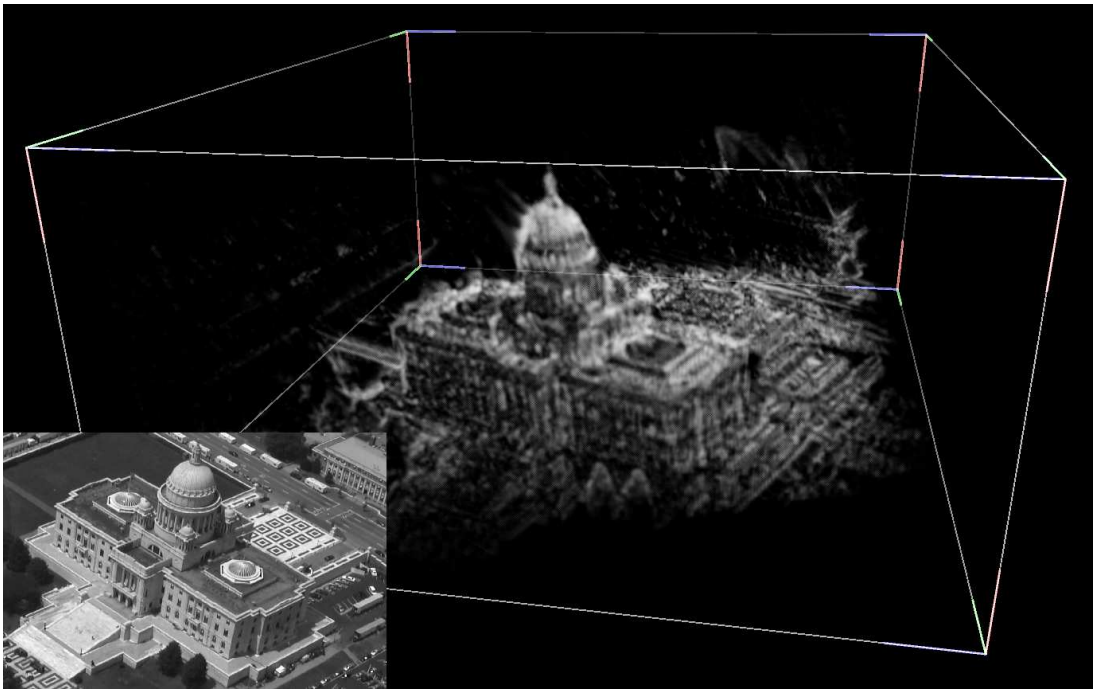

Figure 4.13: A volume rendering of the recovered voxel geometry from the capital building sequence. White voxels have higher probability.

4.2.1 Misalignment Error

The frequent false detections along sharp edges observed in the previous section are inherent to the discretization of the world into voxels. In general the projection of the voxel center in each image will not be perfectly aligned with pixel of the surface it contains. One reason that this misalignment occurs is that the camera models for each image may only be accurate to within a pixel or so, meaning the voxel projection in an image may lie within a half-pixel radius of its true projection. Another reason is that the surface may not lie exactly in the middle of the voxel, so that even with perfect cameras the projection of the voxel center may not hit the same part of the surface in the image every time.

For the majority of surface voxels this alignment error will not cause too much variation in the observed pixel intensities because the majority of surfaces in the world, especially in aerial imagery, have uniform or slowly varying texture. In this case, even when voxels are misaligned with pixels in the image the range of intensities seen will still all be very close to the same value. However misalignment can cause problems at surfaces which have sharp transitions in pixel intensity, such as road markings.

Figure 4.14 shows histograms of pixel intensity along some edges in a sample image. The observations are nearly uniformly distributed between the two intensities on either side of the edge, though the histogram is a bit noisy due to the finite number of samples. These histograms indicate two things. First, the observed uniform distribution, especially for sharper edges, is similar to what one might expect to find in the appearance model of a non-surface voxel, meaning it will be harder to determine whether these edge voxels are surface. Secondly, even if the edge voxel is learned as part of the surface, its learned appearance model will give low probability to all observed intensities and hence produce more false detections.

These alignment errors, while significant, can be overcome. Similar errors are common in mosaic-based change detection algorithms for pan-tilt cameras, where misregistration of a new image to the mosaic can cause errors along edges. There has been work done to correct these errors such as Farin and de With [11], who determine which pixels are at "high-risk" of being mistaken for change and remove them from the final change detection map. Adapting this or another similar technique to the VAM framework is a

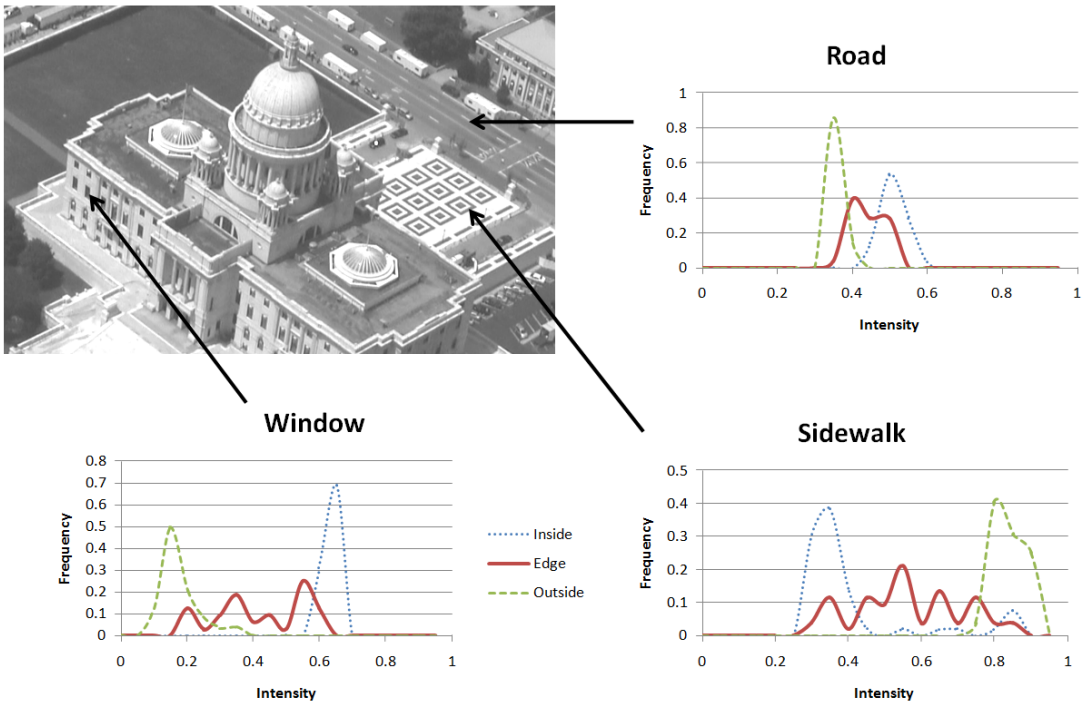

Figure 4.14: Histograms of observed intensities in neighborhoods around pixels on sharp edges. Samples were obtained by taking all pixels within 1 pixel of a line along the indicated edges. The solid graph for each example shows the observed intensities along each edge. The dashed graphs show the observed intensites in the flat regions to either side of the edges for comparison. The histograms on the edges are fairly uniformly spread over the area between the border intensities, though there may be dips due to low sampling. These histograms indicate the level of variation one could expect to see in a voxel on one of these edges.

straightforward problem.

4.2.2 Non-Lambertian Surfaces

Implicit in the VAM framework is the assumption that surfaces in the world appear the same when viewed from any direction. Surfaces with this property are called Lambertian and their observed intensities are determined by the equation:

$$I = I_a k_a + I_d k_d (N \bullet L) \tag{4.2.1}$$

where the observed intensity I is a function of the ambient I_a and directional I_d lighting intensities, ambient k_a and directional k_d reflection coefficients (color of the surface), and the angle between the surface normal N and the lighting direction L.

However many surfaces encountered in real urban scenes do not have this Lambertian property including car windows, metal roofs, and water. These shiny surfaces have specular reflections when viewed from certain directions which appear much brighter than the usual Lambertian color. These specularities can range in intensity and frequency of occurrence from the case of water, where specularities occur only in rare viewing directions but are bright, to tar roofs which have a specular highlight from many viewing directions yet are never very intense. These rarer specularities can cause false detections in VAM. Though the viewing direction, lighting direction, and surface normal have to line up in just the right position for specularities to occur, with all of the different surfaces in a scene the odds are that at least a few surfaces will have bright specular highlights in a given image. Any attempt to remove these specularities in a post processing step is not ideal as any such technique is likely to bias the change detection results against all white objects in the scene.

Figure 4.15 shows an example of a false detection on a metal roof due to specularity. The first time the specularity is observed it causes a gross error, but after training on more images containing the specularity the false detections are less extreme. This example raises the question of how well the mixture of Gaussians appearance model is able to "learn" the specularity.

This question is impossible to answer in general since the ability to learn the specularity

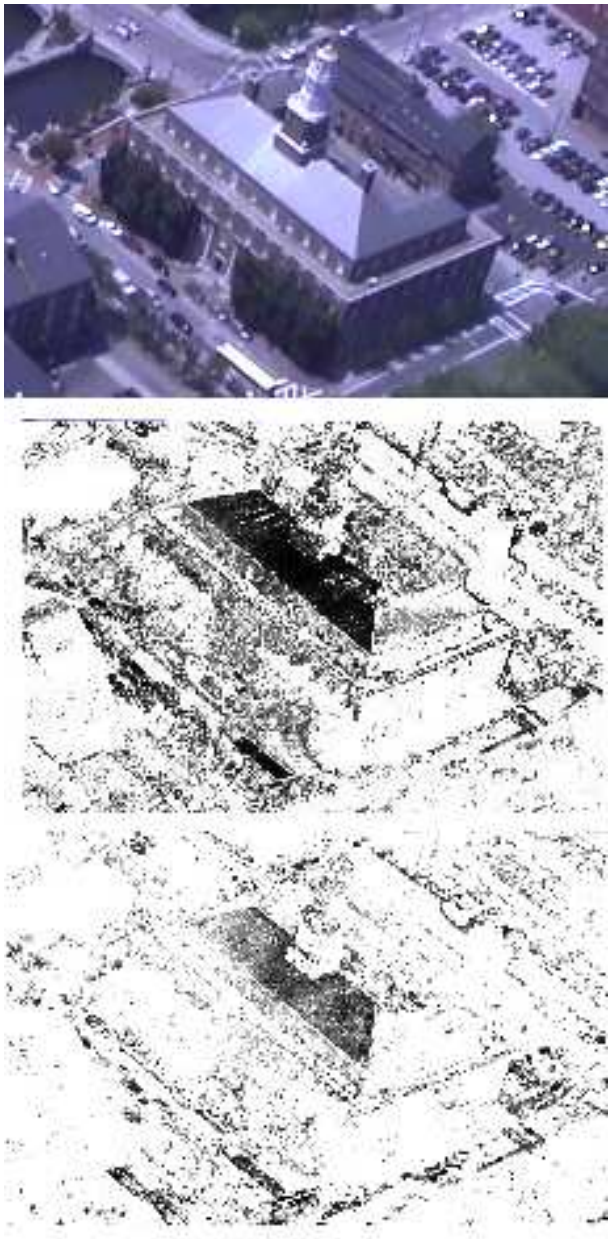

Figure 4.15: False detections are made on this specular metal roof when viewed from certain directions. The errors are especially bad when first observing the specularity (middle), however after training on a wider range of viewpoints the specularity has higher probability (bottom).

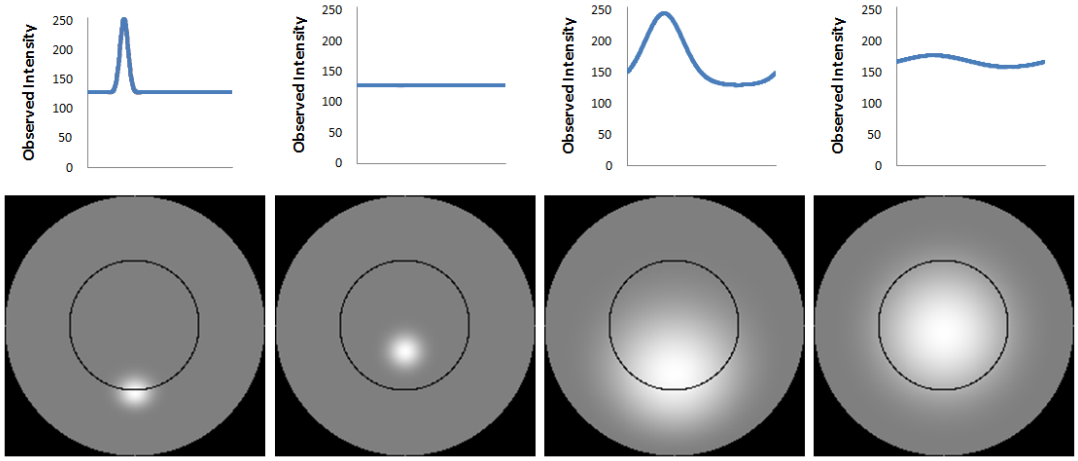

Figure 4.16: Sample observations of several synthetic specular surfaces viewed from a circular arc overhead. Each bottom figure shows a surface seen from all points on a viewing hemisphere, with the black circle indicating a special viewing path. The graphs above show the observed intensities along this viewing path. Depending on the surface orientation and specular radius the variation in intensities may be extreme or non-existent.

is completely dependent on the type of surface as well as range of camera viewpoints of the scene which may be arbitrary. In the context of aerial imagery, the flight path could be a straight line, circular arc, figure 8, or arbitrary zig-zag path and as such a specular highlight on a particular surface may be observed frequently, infrequently, or not at all. Figure 4.16 shows some synthetically generated observations of a specular surface from a circular flight path. The range of observed intensities depends entirely on the orientation of the surface and the magnitude of the specularity. For instance the first flight path hits a small specular highlight once, which will be difficult for a mixture of Gaussians to learn as background given its rarity. However there is no problem in the second example as the flight path misses the specularity entirely. The third and fourth examples have a much broader specularity, which may be difficult to learn in the third example since it causes so much variation, but not as difficult in the fourth example as the flight path and surface are accidentally aligned so that the variance is small.

Little can be said in the general about the ability of a mixture of Gaussians to learn the appearance a specular surface because of this dependence on observation viewpoints. However it will be insightful to consider the case where viewpoints are uniformly dis-

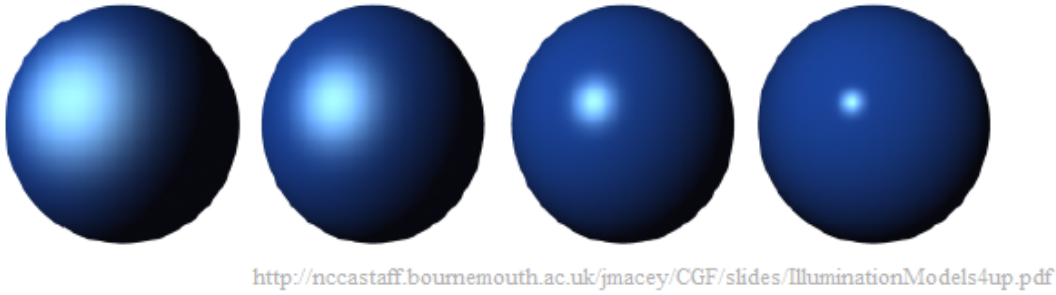

Figure 4.17: Examples of the Phong model with α=8, 16, 64 and 256.

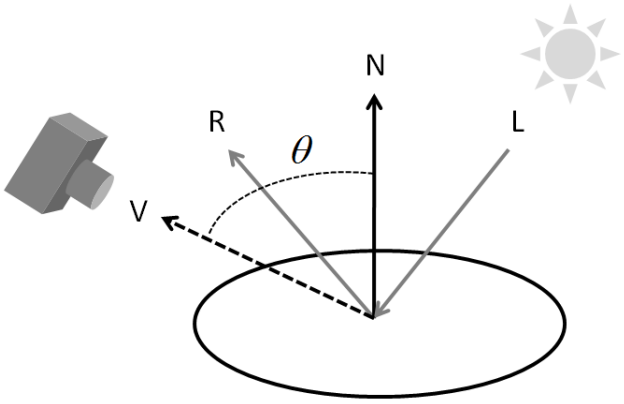

Figure 4.18: Variables in the Phong model.

tributed on the viewing hemisphere. There may be real data collection scenarios which approximate this condition and with a sampling of intensities spread out over many viewpoints there is some chance that a good model of the specularity can be learned. To make an analytic solution tractable, suppose that the specularity can be modeled by a simple Phong model [31] [13], where equation 4.2.1 is augmented with a specular term:

$$I = I_a k_a + I_d[k_d(N \bullet L) + k_s(R \bullet V)^{\alpha}] \tag{4.2.2}$$

where k_s is the specular-reflection coefficient, V is the viewing ray, R is the reflected light ray, and α is a coefficient which determines the radius of the specularity. These variables are illustrated in figure 4.18. Some sample spheres rendered with a Phong model with different values of α are shown in figure 4.17. To simplify the situation further suppose that the normal vector points straight up and the lighting vector points straight down so

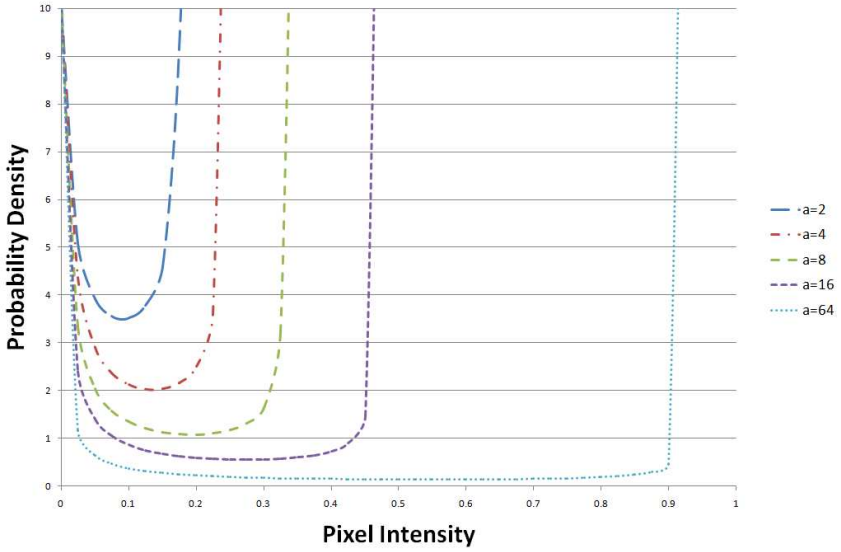

Figure 4.19: The density function from equation 4.2.4 plotted for various values of α.

that equation 4.2.2 reduces to:

$$I = I_l + I_s cos^{\alpha}\theta \tag{4.2.3}$$

where I_l and I_s are the Lambertian and specular intensities for this fixed lighting and θ is the angle of the viewing direction V to the normal. Now suppose that the view angle θ is uniformly distributed on $[0, \pi/2]$. This induces a probability density on I which one can verify is:

$$p(y) = \frac{2}{\pi}\frac{1}{\alpha I_s}\frac{(y/I_s)^{\frac{1}{\alpha}-1}}{\sqrt{1-(y/I_s)^{\frac{2}{\alpha}}}} \tag{4.2.4}$$

on the range $[0, I_s]$ assuming that $I_l = 0$ for convenience. This density is plotted in figure 4.19 for various values of α. Note that the Phong model is not a physical model and in particular no conservation of energy holds for different values of α. To compensate for this, I_s is adjusted for each α such that: $\int_0^{\pi/2} cos^{\alpha}\theta d\theta = C$ for some constant C. Because of this the plotted densities have different domains. The densities are asymptotic at 0 and I_s and most of the probability is at near 0. This probability increases as α increases with $P(I < .1) = .18$ for $\alpha = 2$ and $P(I < .1) = .77$ for $\alpha = 64$. This explains why the curves

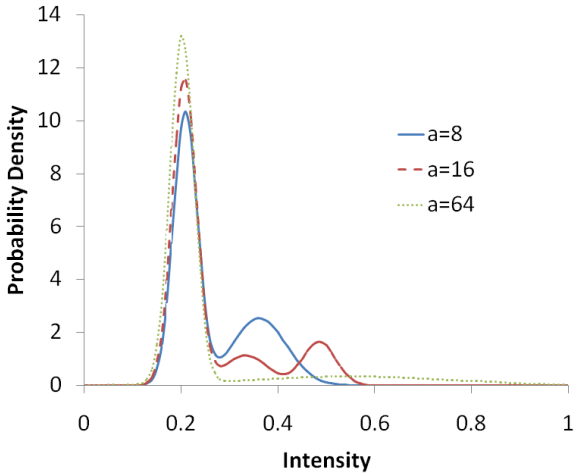

Figure 4.20: Density functions for mixtures of Gaussians trained on simulated Phong observations for different values of α. All learned distributions have a peak at the Lambertian color 0.2 and the maximum specular value which varies. The specular peak for $\alpha = 64$ is at 0.8 but has so little weight in the mixture that it is almost unnoticable.

for higher α appear to have less area under the curve, when in fact they all integrate to 1.

One notices a few things about these curves. Firstly the fact that the distribution is bimodal means that for uniformly distributed viewpoints at least the mixture of Gaussians will be a good fit to the data and that the Lambertian and specular extremes will be the primary modes of the distribution. Secondly for low α surfaces, it is likely that the range of intensities between the Lambertian and specular ends will be correctly classified as background since this range is small and has high probability. For higher α, the huge range of low probability intensities between the Lambertian and specular ends are likely to be misdetected as change for any reasonable probability threshold in the change detection. However in this case these intensities appear infrequently and shouldn't contribute much error to the results. For intermediate α, how much of the middle range is classified as background will depend heavily on the choice of probability threshold in the change detection.

In the final theoretical experiment a mixture of Gaussians was trained simulated data from Phong models for different values of α. The plots in figure 4.20 have peaks rather than the asymptotes in figure 4.19 but they occur in the same places. Next, the ability of these learned mixtures to distinguish observations coming from the Phong model and

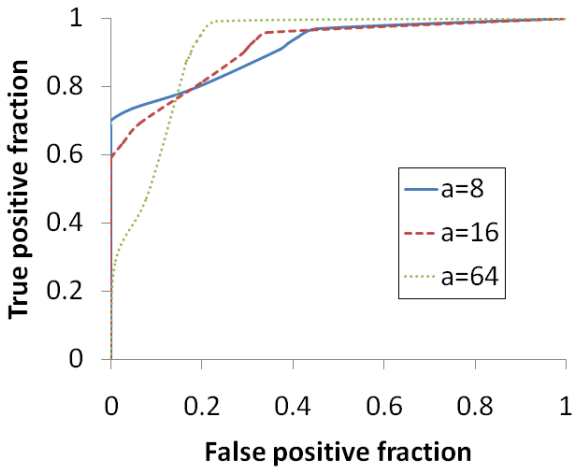

Figure 4.21: ROC curves comparing the fraction of correctly detected changes (uniform distribution) to the fraction of false alarms (Phong distribution).

observations coming from a uniform distribution was tested. The uniform distribution models actual changes which could be conceivably any color, so this gives an idea of how well the learned mixture distribution will be able to correctly identify changes. ROC curves are given in figure 4.21 which plot correctly detected changes (from uniform) against false detections (from Phong) at different probability thresholds. For low alpha, i.e. mostly Lambertian, surfaces it is easy to detect the majority of uniformly-distributed changes, however detecting the remaining changes results in many false alarms, since these observations will be similar in color to the specular appearance of the surface. For high alpha, detecting the majority of changes requires making the occasional false detection on the rare specular observation, but once one is prepared to accept these errors one can obtain near perfect detection of changes. These results suggest that learning the full appearance of slightly specular surfaces is possible and will not result in many missed detections, but that the specular component of very shiny surfaces surfaces cannot be learned since it is rarely observed.

4.2.3 Sampling Deficiency

In addition to misalignment and specularity errors which occur when the VAM learning fails, there can also be misdetections when there insufficient sample images of a surface. A

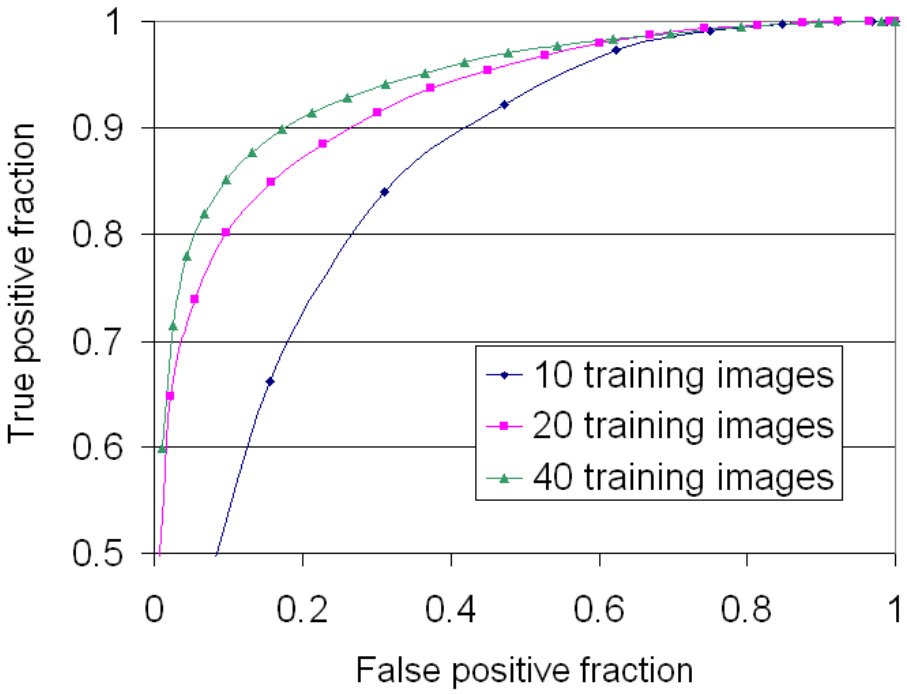

Figure 4.22: Change detection ROC curves for VAM applied to the plasticville sequence with different numbers of training images. 20 images is sufficient to produce good change detection, with more images providing marginal improvement.

surface needs to be seen multiple times from multiple directions before its geometry and appearance can be learned, and until enough images are seen the object may be labeled as foreground. For example a helicopter taking images of a building may move so that a new side of the building is revealed for the first time. This side is "new" and so will be marked as change for a few images. In another example a parked car leaves its parking spot, which is marked as change for several images until the ground surface is learned. To some extent this is a philosophical problem since it hinges on the question of how many times a surface needs to be seen before it is considered "background", however these "errors" may decrease the usefulness of the change detection in a real application.

Like specularity errors it is hard to say anything in general about the error rates due to insufficient observations since they depend on many factors including: how many images the scene has been trained on, current occupancy probabilities, the learning rates of the

mixture models, and the homogeniety of region. It is however possible to investigate how the number of training images effects the overall error rate. In an experiment on the "plasticville" model town sequence, VAM is trained separately three times on the image set with a different number of training images in each run. ROC curves are given in figure 4.22. As seen, 20 images is enough to produce highly accurate change detection results with additional images providing marginal improvement. With 10 images some parts of the scene have not been adequately learned and there are more false detections.

4.3 Conclusions

The VAM framework has been shown to produce accurate change detection results on many real-world image data sets. The true changes in these images are limited to moving vehicles, but VAM is able to detect them with a limited number of false alarms. The limited errors that do occur are not entirely unexpected. Errors caused by misalignment have been known to occur in registration-based algorithms and techniques to cope with them exist. Specularities are rare and unavoidable errors but can be partially learned.

What is important to note is that none of the errors are caused by incorrectly learned 3-d geometry. The online learning algorithm presented in chapter 3 is able to estimate an accurate 3-d model of the scene as the theory says it should. This is shown by lack of errors due to occlusion in the change detection results and by visual inspection of the voxel occupancy probabilities. The appearance of this surface has also been properly learned as seen in the sample image-based renderings. The experiments in this chapter back up the theory in chapter 3 and demonstrate that the VAM framework not only works, but produces highly accurate change detection results on real image data.

Chapter 5

VAM for Satellite Imagery

The VAM change detection framework presented in the last few chapters is immediately usable for many applications. The algorithm presented so far makes the assumption that the appearance of surfaces is constant or slowly varying between images, just as in the Stauffer-Grimson algorithm. For scenarios like an aerial surveillance platform, UAV video, or an industrial inspection system this assumption holds and VAM will work well. However one application of great interest is change detection in satellite imagery, which does not meet this requirement.

As seen in figure 1.1, the appearance of a surface in satellite images taken at different times can be quite different. Time of day affects the lighting and shadows of the scene, and haze in the atmosphere can decrease contrast in some images. The mixture of Gaussians appearance model currently used in VAM is not flexible enough to handle variation from these sources. Still, aside from this simple appearance model the VAM change detection framework is well suited to handling the complicated 3-d scenes observed in satellite imagery.

This section presents modifications to the proposed VAM framework to allow it to handle the additional variation encountered in satellite imagery. First the appearance model is upgraded to a lighting-dependent version, which uses the known position of the sun to make better decisions about normal and abnormal appearance. Secondly, a preprocessing normalization step is applied to compensate for global changes in brightness and contrast due to haze and varying sun intensity. Aside from these modifications, the

online updating and change detection equations for VAM in chapter 3 are unchanged.

The lighting dependent appearance model and intensity normalization processes are described in the next two sections. Additional experiments on the extended VAM framework follow in the third section. These experiments are run on a set of satellite images of Baghdad, Iraq taken over the years 2002-2007, on which interesting and subtle changes are detected.

5.1 Local-Lighting Appearance Model

The presence of moving shadows and variable illumination in satellite images means that the mixture of Gaussians appearance model used in VAM is no longer sufficient. If the lighting was varying slowly between images then the mixture of Gaussians would be able to adapt to the changing intensities, just as it does in the standard Stauffer-Grimson approach. However satellite image collection is sporadic, so that during the online updating of VAM the lighting direction in one image may be very different from the lighting direction in the previous image. The solution is to use a lighting-dependent appearance model so that the intensities in images with very different lighting directions are never directly compared.

The ideal lighting-dependent appearance model would be an approximation of the bidirectional reflectance distribution function of the surface material in a voxel. However accurately estimating a parameterized BRDF in an online algorithm is difficult, especially in the presence of occlusion and shadows. In addition, it frequently happens that the entire observed range of lighting conditions is just a small portion of the lighting sphere, making estimation of the BRDF numerically unstable. For example, satellite imagery is most commonly collected by sun-synchronized satellites in the early morning so that the range of lighting directions over the year is only about a quarter of the full daylight range of the sun. Given these complex phenomena, a non-parametric appearance model for observed illumination directions is a better choice than a parameterized physical BRDF model. A non-parametric function can model appearance locally around each lighting direction. The main disadvantage of learning appearance locally is that many images will

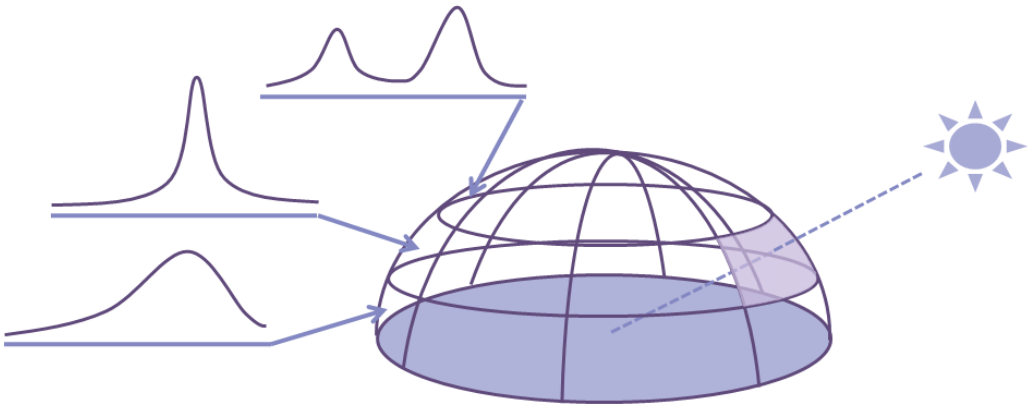

Figure 5.1: The space of lighting directions is discretized with each bin containing an independent mixture of Gaussians appearance model.

be needed to robustly learn each local appearance. However, using a local model is the only way to guarantee that local effects like shadows and mutual illumination are modeled accurately.

The method proposed here is to discretize the space of illumination directions and model each lighting direction cell independently with a mixture of Gaussians as in chapter 3 (see figure 5.1). The idea is that only Gaussian mixtures trained on images with similar lighting directions will be used to compute the probability of a new image. Choosing the discretization of the lighting directions is application dependent, with finer grid cells producing more accurate results but requiring more images to learn each mixture model. For the Baghdad satellite images used in this paper, three illumination direction cells were sufficient to produce consistent appearance models with the range of lighting directions present.

5.2 Intensity Normalization

The lighting-local appearance model just described accounts for changes between images caused by lighting direction, but variations in light intensity and atmospheric haze must also be corrected before the algorithm processes an image to detect changes. Narasimhan and Nayar [30] present a model for the pixel intensity observed in an outdoor image:

$$I = I_\infty \rho e^{-\beta d} + I_\infty (1 - e^{-\beta d}) \tag{5.2.1}$$

Where I_∞ is the sky intensity, ρ is the normalized radiance of the surface, β is the scattering coefficient of the atmosphere, and d is the distance from the surface to the observer. It follows that two observations I_1 and I_2 of the same surface intensity ρ:

$$I_1 = I^1_\infty \rho e^{-\beta_1 d_1} + I^1_\infty (1 - e^{-\beta_1 d_1}) \tag{5.2.2}$$

$$I_2 = I^2_\infty \rho e^{-\beta_2 d_2} + I^2_\infty (1 - e^{-\beta_2 d_2}) \tag{5.2.3}$$

can be related by a linear relationship after solving for ρ:

$$I_1 = g \bullet I_2 + off \tag{5.2.4}$$

where the gain g and offset off are dependent only on global image parameters so this relationship also holds when I_1 and I_2 are full images. The unknown gain g and offset off can be estimated if enough pixels from I_1 can be matched to pixels in I_2. The VAM framework which is already in place provides the necessary constraints to find optimum values of these parameters. Suppose the model has been trained on several images so that the 3-d structure and surface appearance of the scene has been learned, then the correct gain and offset of a new image I_j can be estimated by maximizing the image probability after normalization:

$$P(norm(I_j)) = \frac{1}{|I_j|} \sum_{y \in I_j} P(g \bullet I^y_j + off) \tag{5.2.5}$$

This solution should clearly estimate the correct normalization when the 3-d structure of the world has been learned, since pixels in I_j are being compared to the learned surface appearance, which is a good approximation to the true appearance. What is surprising is that this maximization will work well even when only one image has been trained on, i.e. before any 3-d geometry has been estimated. In this situation, the probability distribution for each pixel given in equation 5.2.5 is just a distribution of all intensities seen along an epipolar line in the first image. Fortunately, aerial scenes are typically made up of large

areas of mostly uniform intensity, i.e. roads, roofs, grass. Thus, the distribution will typically have a peak around its intensity from the first image even though the intensity from the same corresponding 3-d location contributes very little to the distribution. The accuracy of this normalization process when VAM has been trained on few images is examined in a later section. An efficient method for calculating the optimal gain and offset is given in chapter 7.

5.3 Experiments

This section presents additional experiments on the VAM framework, using the satellite extensions presented in this chapter. Most experiments in this section are carried out on a collection of Quickbird satellite images of Baghdad, Iraq taken over the years 2002-2007. These images have a typical range of viewing and lighting directions for satellite images, displayed in figure 5.2. The satellite images are also very large, often exceeding 2GB in size and each covering an average of 100 square miles at .66m pixel resolution. Several smaller regions averaging 1 square mile in size were cropped out and used for experiments. These images were associated with cubic rational polynomial camera [18] models, which were observed to be misregistered by an image translation and were corrected manually using a single image point correspondence with a known 3-d point. This correction step is simple enough that it could be conceivably automated in the future.

Though the Baghdad area is one of the most frequently imaged locations on the planet, the number of available Quickbird images of the region pushes the lower limit on how many images are needed for VAM to perform well. Satellite collection is typically done with infrequent return visits so that the satellite can be used to collect imagery for a wider area. The greatest region of overlap in the 200 or so available Baghdad images is about 32 images. Though it is shown in figure 4.22 that VAM can produce good results with as few as 20 images, with a local-lighting appearance model more than 20 images are typically needed to properly learn the appearance under all lighting conditions. As such the results in this section are good, but would be better if more images were available. In actual use there should be no shortage of images, since collections from many different satellites can

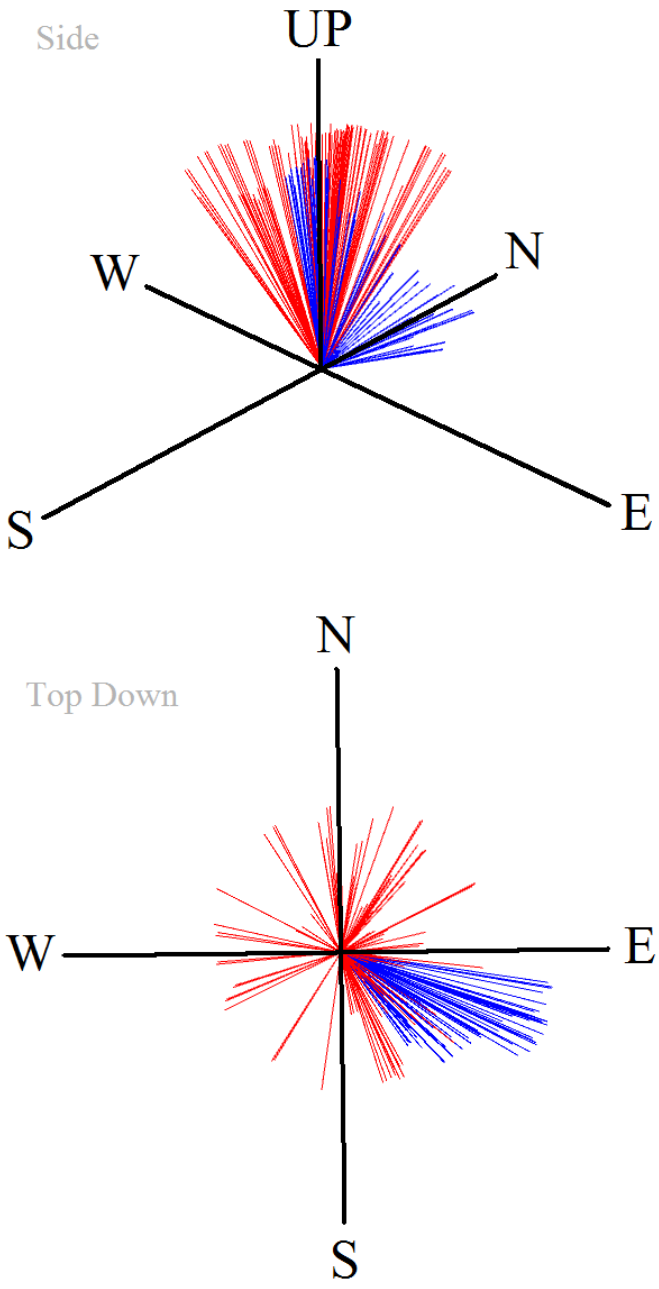

Figure 5.2: A plot of the range of lighting and camera directions present in the Baghdad satellite image collection. Views are shown from the side and top-down with red lines for camera directions and blue lines for lighting directions. The camera directions are fairly uniformly distributed across the top of the sphere. Since the satellite is sun-synchronized all images were taken at around 7:45 AM and as a result the lighting directions are concentrated in a narrow strip to the southeast.

be pooled together assuming they have similar imaging systems. Different satellites may have different collection times, leading to more illumination bins which must be modeled, however most surveillance satellites take images in early morning or late afternoon so there will be some overlap between different satellites.

It is difficult to quantify the performance of VAM for scenes in satellite images. Unlike the helicopter footage used in chapter 4, where the only true changes were moving vehicles, the concept of a true change is harder to define when considering longer term structural changes to the scene. For example when a new building is constructed its first observation should be considered a change, but how many more images of the building most be taken before it is considered "background"? Certainly one observation is not enough for VAM to learn the new building's structure and appearance while another 30+ images would definitely be enough. But is it correct to mark a building as background if it has only been around for 5 images? 10 images? When evaluating our algorithm with experiments it was decided to consider an object as background immediately on its second observation rather than devise a subjective criteria for how many observations are needed. Performance curves will be lower for this conservative choice when new structures are not observed enough times before evaluating a test image. Ultimately these "errors" shouldn't be a problem in application, as a complete system will have application-dependent criteria for when to ignore areas of recent change.

For all of the tested Baghdad scenes, a local-lighting VAM was trained on 27-29 images taken during the years 2002-2006. Changes were detected in 3-4 images from 2007, with ground truth marked by hand relative to the last image of 2006. The lighting directions were grouped into 3 lighting bins for the local-lighting appearance model such that each bin contained 9-10 images. Using more bins would be desirable but impossible with this few number of images.

For a few sequences, the results from VAM are compared to Carlotto's algorithm [6] (described in chapter 2) which uses a global affine warp to align two images and detects pixel changes using a joint probability measure. The affine warping was done using 50-150 manual correspondences per image pair, though this could be automated at the cost of additional error. Each test image was compared to the ground truth reference image,

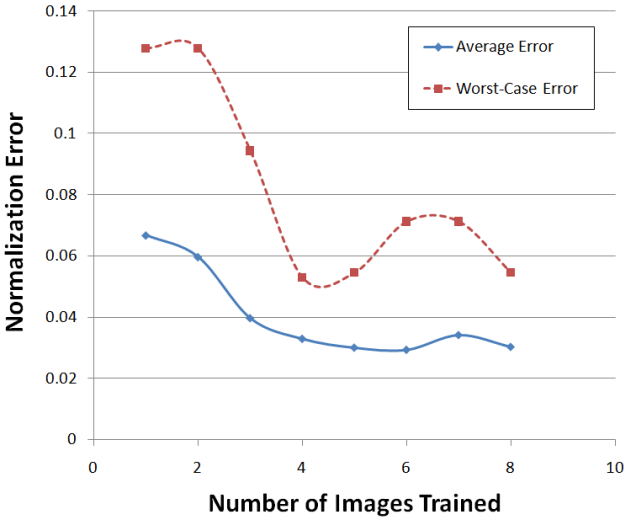

Figure 5.3: Intensity differences between an image and its normalized version after correcting for artificially added haze. Errors are quite low even after training on just one image in both average and worst-case errors for a set of 8 images. (Error is expressed in percent)

which gives the joint probability technique an advantage over VAM for these experiments since it takes VAM several images to learn new structures. However the joint probability technique will do poorly whenever there is significant 3-d relief between images since it has no compensation for 3-d structure.

5.3.1 Normalization Accuracy

Before testing the extended VAM framework on satellite images, the accuracy of the proposed intensity normalization technique is evaluated on a controlled data set. The normalization technique is tested on a set of 8 images of the model "plasticville" town taken under the same lighting but different viewing angles. Haze was artificially added to each image using equation 5.2.1. VAM was trained on one un-hazy image at a time and after each the normalization algorithm was run on all 8 hazy images. The average pixel intensity difference between each normalized image and un-hazy image was calculated and a plot of worst-case and average errors over the 8 images is given in Figure 5.3. The results show that the normalization algorithm can recover the original image intensity with an average 3% error once some 3-d structure has been learned and a reasonable 7% average

error when no geometric structure is known.

5.3.2 Hiafa St.

The Hiafa St. scene was the first area selected for testing because it has some of the tallest buildings in Baghdad which often cast long dark shadows across the neighborhood. This sequence tests how well the local-lighting appearance model is able to cope with moving shadows. VAM performs very well on this sequence as shown in figure 5.4, especially compared to the joint probability algorithm which fails because the affine warping is unable to cope with off-planar 3-d structure. Most of the true changes in the test images are vehicles, which are both not very interesting and hard to detect because they are only a few pixels across and tend to blend into the road at this resolution. As a consequence the performance ROC curve for VAM in figure 5.5 is only able to detect about half of the true changes in the scene with an acceptable amount of false alarms. Still when more interesting structural changes to the scene occur VAM is able to pick them out easily. Some of the more interesting changes detected during training are shown in figure 5.6. Finally a volume rendering of the successfully recovered 3-d structure is shown in figure 5.7.

This area of Baghdad was chosen in part to test the ability of the local-lighting appearance model to learn long moving shadows particularly when the lighting angle is low. For low lighting angles a small change in lighting direction can move shadows quite far, increasing the variance in observed intensities for those light bins. The results show that the light binning is able to manage in this respect since there are few false alarms on the shadows. However for low light angles there are occasionally false positives on some sun-facing walls as shown in figure 5.8. These false detections occur because the amount of reflected light on these walls varies considerably from image to image, possibly due to the amount of atmosphere sunlight must travel through at these low angles.

5.3.3 Orchard

Though VAM has worked well on dense urban scenes, it is unclear how it will perform in the absence of man-made structures. Having buildings and roads of different colors is what

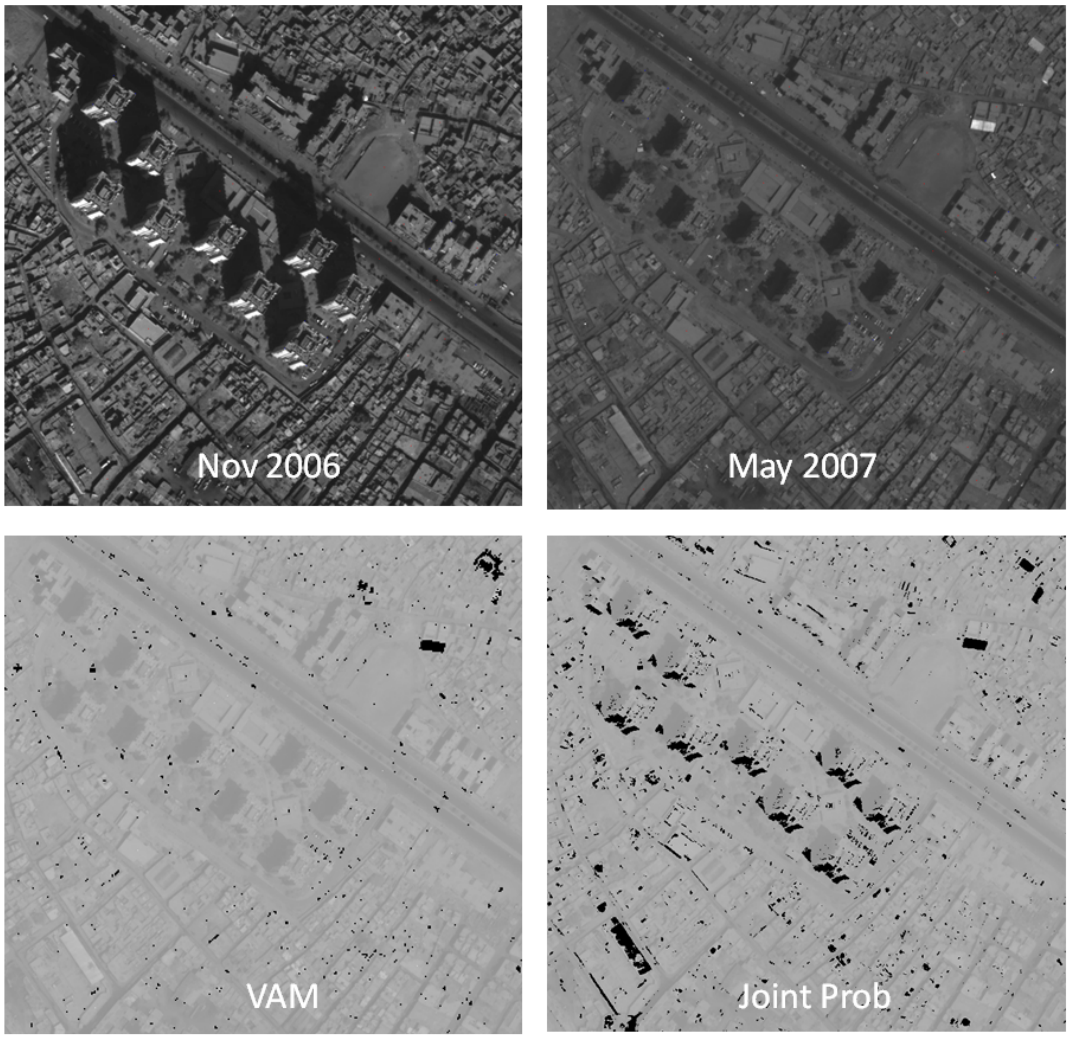

Figure 5.4: Example detections for the Hiafa St. sequence. Top row shows before and after images of the scene, while bottom row shows change detection results from VAM and joint probability. Most of the detections in VAM are vehicles with some interesting structural changes in the upper right corner. The joint probability technique makes significant errors on the sides of the tall buildings since it doesn't model 3-d structure.

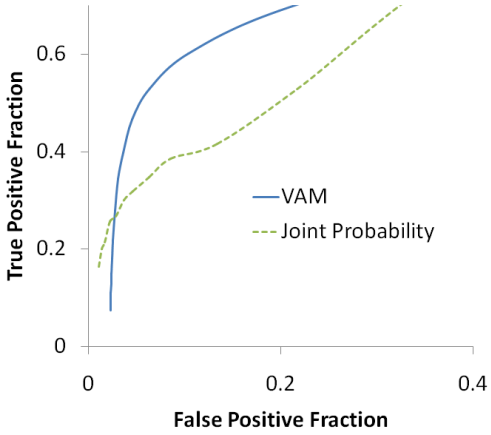

Figure 5.5: ROC curves for the Hiafa St. sequence. VAM is able to detect about half of the vehicle pixels in the scene with an acceptable false alarm rate. The joint probability technique makes 2-3 times as many false detections for any reasonable percentage of correct detections.

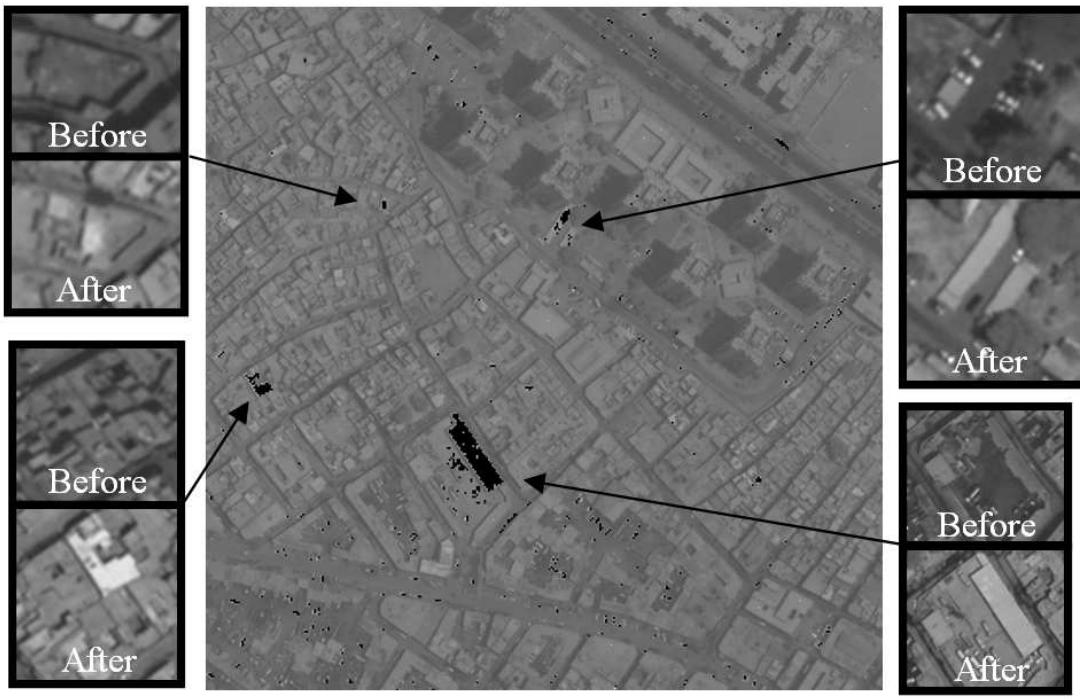

Figure 5.6: Some interesting structural changes in the Hiafa St. scene detected in a training image.

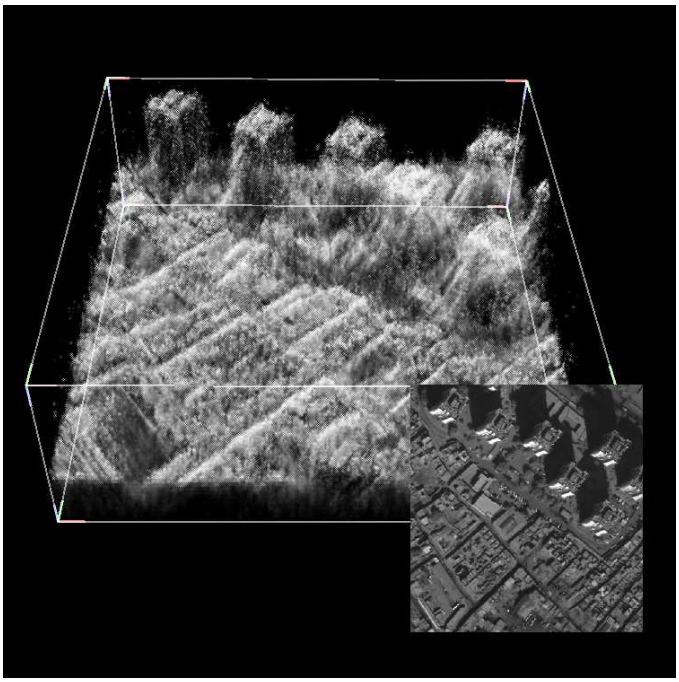

Figure 5.7: A volume rendering of the recovered voxel geometry from the Hiafa St. sequence. White voxels have higher probability.

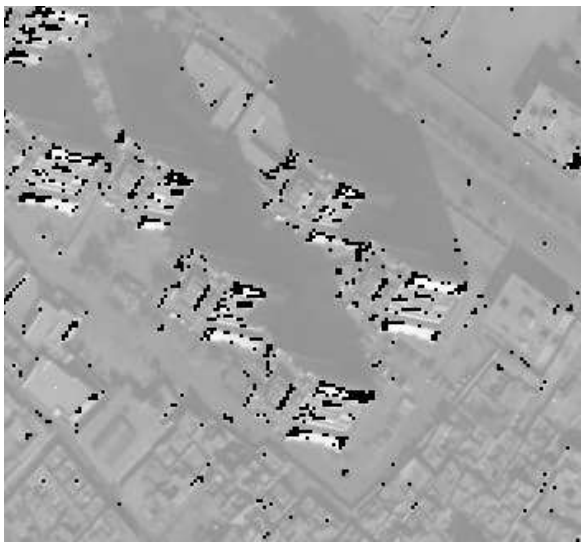

Figure 5.8: False positive detections in the Hiafa St. sequence on sun-facing walls.

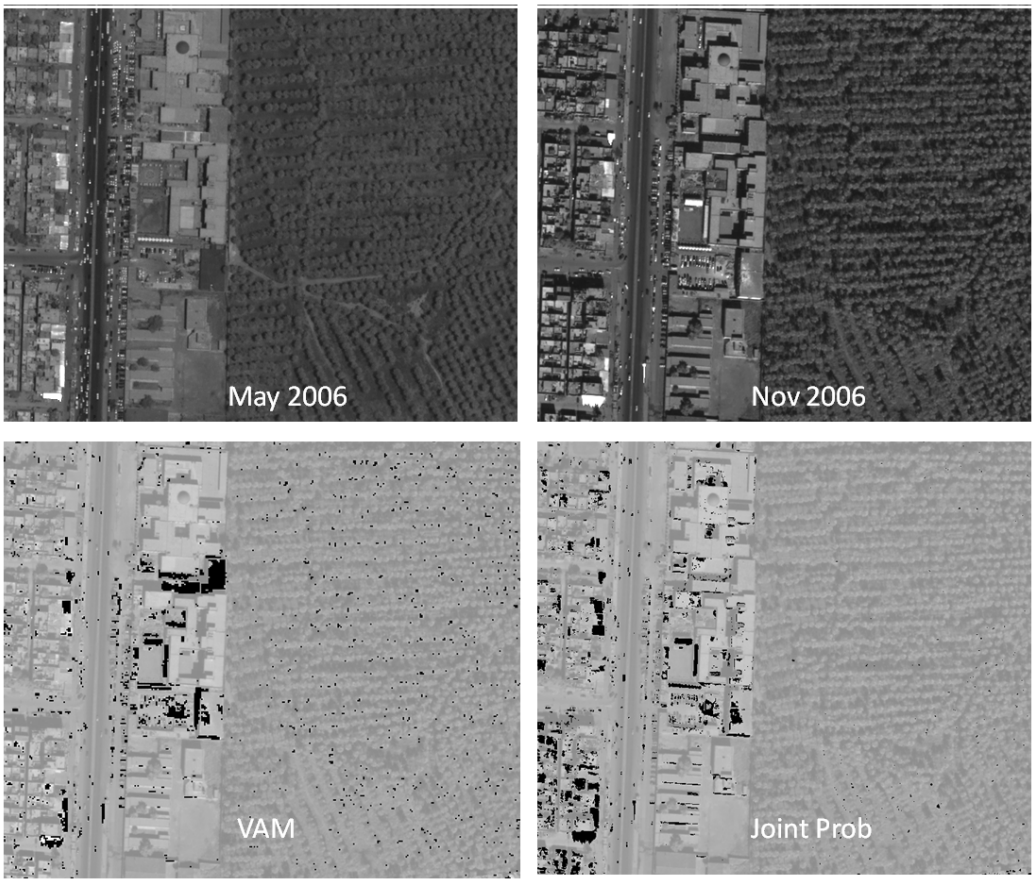

Figure 5.9: Example detections for the orchard sequence. Top row shows before and after images of the scene, while bottom row shows change detection results from VAM and joint probability. The changes detected by VAM on the right side of the street contain most of the true change to the scene. There are very few true changes on the left side of the street where the joint probability produces many false detections.

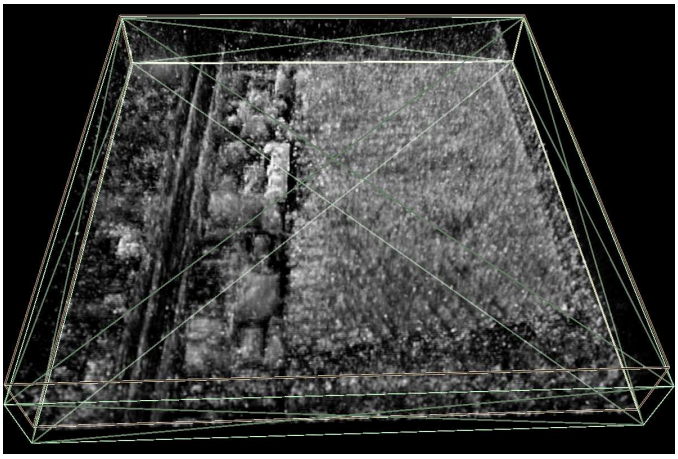

Figure 5.10: A volume rendering of the recovered voxel geometry from the orchard sequence. White voxels have higher probability. The orchard region has many voxels with relatively high occupancy probability in the empty air above the trees, unlike the neighboring urban area.

allows VAM to estimate the 3-d structure of the scene, and in a scene that is mostly trees there may not be enough information for a reliable reconstruction. So a neighborhood in western Baghdad containing an orchard was selected for testing to see how well VAM could handle a large area of vegetation. Some sample images and change detection results are shown in figure 5.9.

There are scattered small false detections in the orchard area but no large errors. A volume rendering of the recovered geometry in figure 5.10 gives some insight into the cause of these small errors. Though the 3-d geometry of the trees have been learned reasonably, there are many more noisy voxels with high probability in the space above the orchard region than the neighboring urban area. This noise is caused by the same reconstruction ambiguity that plagues 3-d reconstruction algorithms in textureless regions. However the lack of false detections here highlights one of the key strengths of VAM, the ability to obtain accurate change detection results even when the 3-d geometry cannot be recovered perfectly. The joint probability technique does reasonably well on the orchard area but makes many false detections on the nearby buildings and misses several important structural changes.

The larger changes detected by VAM in the left half of the image, while not all changes from the May 2006 reference image, are all recently added structures to the neighborhood.

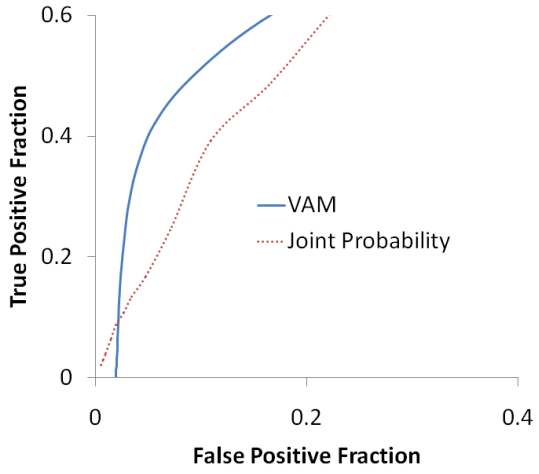

Figure 5.11: ROC curves for the orchard sequence.

Though qualitatively reasonable, those changes not in the reference image are counted as false positives in the error analysis in figure 5.11. As a result the ROC curves aren't quite as high as the Hiafa scene, though the results are visually just as good.

5.3.4 U.S. Embassy

The next experiment on the Baghdad imagery was done at the site of the new U.S. embassy construction site next to one of Saddam's old palaces. Some sample pictures of the scene and results using VAM are shown in figure 5.12. The right lot was empty for most of the image sequence until 2006 when it became an active construction site. All of the new trailer structures in the construction zone are detected while the buildings in the left lot are not. This gives an idea of the kind of large-scale structural changes to an area that VAM is able to detect.

5.3.5 Tigris River

Another potential source of errors in aerial imagery is water. In man made pools and canals there are occasional specularities which can cause false alarms in change detection results. Of more concern in satellite images are rivers which over the course of a year vary dramatically in depth and color. Some sample images of the Tigris River in Baghdad over the years 2002-2007 are shown in figure 5.13. Variations between images include the

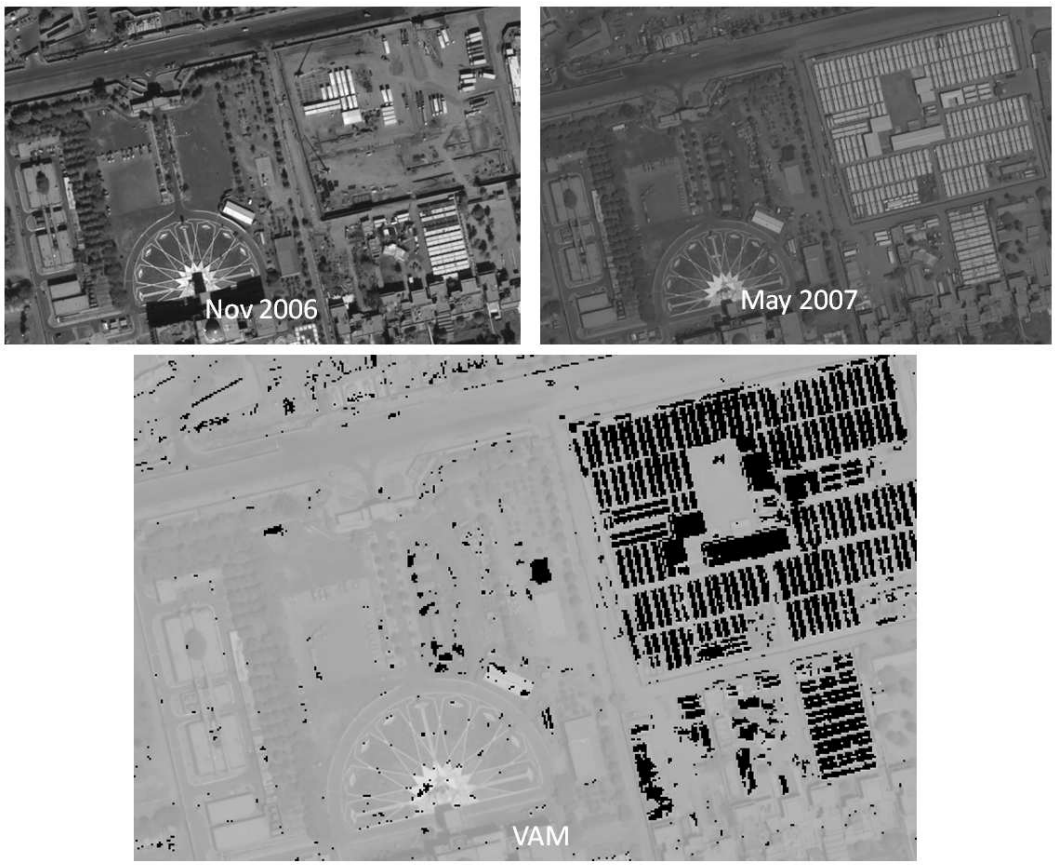

Figure 5.12: Example detections for the embassy sequence. Top row shows before and after images of the scene, while bottom row shows change detection results from VAM. The trailers in the right side construction lot and scattered changes in the left courtyard are all detected with few false alarms.

Figure 5.13: Sample images of the Tigris river over the years 2002-2007.

position of the banks and cloudiness of the water, which presumably varies due to the amount of sediment and pollution present.

VAM was run on 31 images of the Tigris River and some examples of errors made on the water are shown in figure 5.14. In the early and middle images in the sequence huge errors are made when a radical new river appearance is seen for the first time. By the end of the sequence there are fewer false alarms on the river, even for rare appearances. However after learning all these variable river appearances, by the end of the image sequence VAM is no longer able to distinguish all interesting changes in the water. Given the diversity of river appearances, it is unlikely that any technique would be able to correctly identify changes with any degree of accuracy.

5.3.6 Karkh

The last region is a construction site in the Karkh neighborhood of Baghdad which is an area of constant change over the years 2002-2007. A time line of the change construction progress is given in figure 5.15. With the exception of the large far-right building, the entire site is always changing, never appearing the same for more than a few images at a time. This makes it difficult for VAM to detect any kind of meaningful change since there

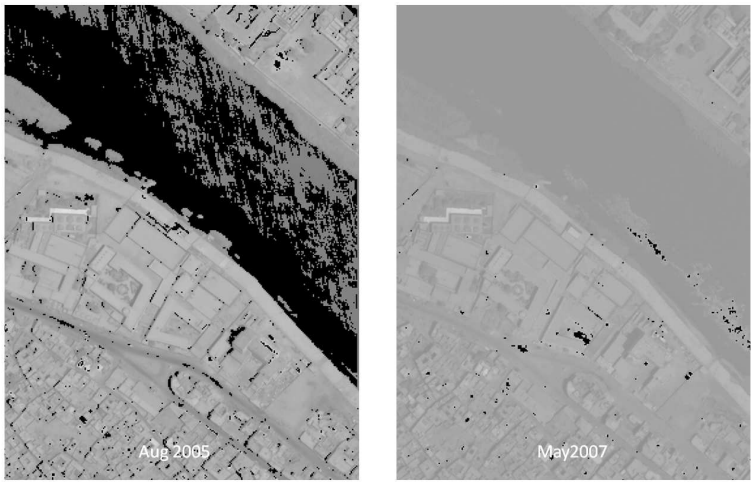

Figure 5.14: Errors in learning the Tigris River. The highly variable appearance of the river causes large errors on August 2005 in the middle of the image sequence. By the end of the sequence in May 2007 enough variability has been introduced into the river's appearance models that they are unable to identify all of the debris floating by.

Figure 5.15: Progress at a construction site in the Karkh neighborhood 2002-2007.

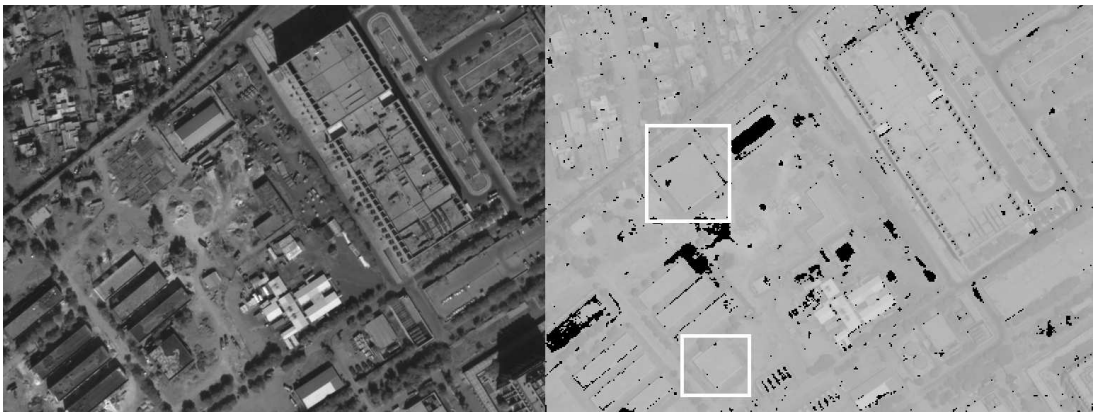

Figure 5.16: Changes detected by VAM in a new image of the Karkh construction site compared to the previous image. In this constantly changing scene it is difficult to determine what the correct classification of foreground and background should be. Here VAM identifies some small buildings and piles of dirt as change, while not making many false detections on the far right building. However it misses two important changes to the scene circled in white.

is no "normal appearance" to learn. Some of the detected changes using VAM are shown in figure 5.16. The changes detected in the construction zone are arguably correct detections and there are few false alarms on the right building. However a few new buildings which appear for the first time in this image are missed because they are in grayscale very close in intensity to the dirt that existed below them earlier in the sequence. The shadows from these new buildings are detected however.

These results highlight two weaknesses of VAM. First is the requirement that many images of an object must be observed before it can be fully learned as background. This is an intrinsic weakness of all background modeling algorithms. The second is the inability to distinguish new objects with the same grayscale intensity as the learned background. This is less of a problem for detecting vehicles as car and road are usually different in color, however it may be an issue for buildings, especially those with textureless roofs. This later problem is unavoidable, and as mentioned before can only hope to be solved by using color information. However shadows cast by new structures are marked as change, so some evidence of the construction is detected.

5.3.7 Predicting Appearance

The local-lighting appearance model introduced in this chapter segregates images with different lighting directions and uses an independent mixture of Gaussians for each lighting bin. One natural question to ask is whether the learned appearance models in these bins can be used to predict the appearance for a new unobserved lighting direction. The answer depends on which lighting bins have been learned and how they are positioned relative to the lighting direction to be predicted. In general if enough of the lighting space has been observed, a higher level BRDF model could be learned on top of the local appearance models. In this chapter we consider the simpler problem of interpolating learned appearance models.

Consider the simplified case where the lighting space is 1-d, i.e. a cross section of the full lighting sphere. The technique presented next can be easily extended when there are samples on the whole 2-d lighting sphere. For lighting bins $l_1 < l_2 < l_3$, if local appearance models at a voxel have been trained for illumination directions l_1 and l_3 then a good initial estimate of the appearance model for l_2 is the following mixture of Gaussians distribution:

$$p^2(y) = \alpha_1 p_1^1(y) + \alpha_2 q^{13}(y) + \alpha_3 p_1^3(y) \tag{5.3.1}$$

where $p_1^1(y)$ and $p_1^3(y)$ are the most heavily weighted modes in the original mixtures of Gaussian for l_1 and l_2 and $q \sim N(\beta\mu_1^1 + (1-\beta)\mu_1^3), \sigma_0)$. So the new appearance is a mixture of its two neighbors most probable pixel intensities and their interpolated intensities. The un-interpolated modes $p_1^1(y)$ and $p_1^3(y)$ are included in the mixture with small weights α_1 and α_3 to boost the probability of shadows in situations where there is a shadow in one lighting category but not the other.

This interpolation scheme was tested on a set of images of a model town taken under different lighting conditions. A directional light was moved in a 180 ° arc over the "plasticville" town and a set of 8 pictures were taken at 10 ° increments. VAM was trained on every other lighting angle and the interpolation method was used to predict the appearance models for the skipped angles. Expected images were rendered and changes were detected for each of the untrained images using the predicted appearance. This process

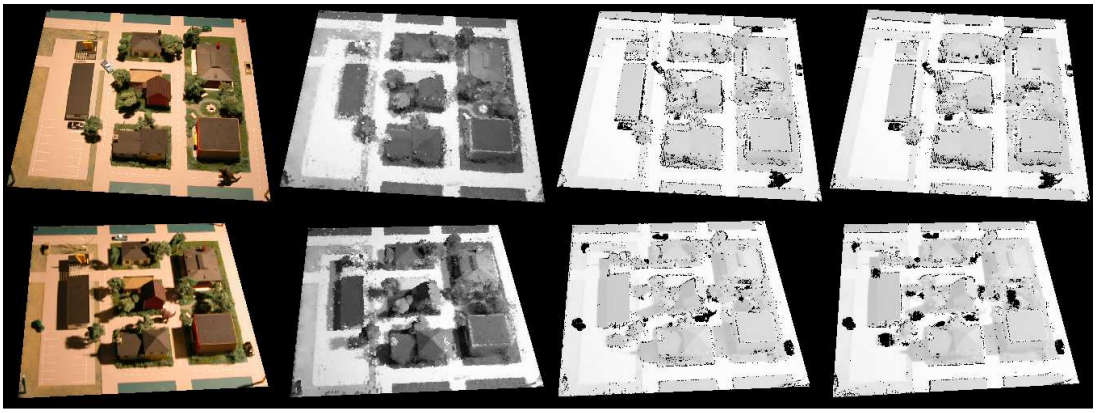

Figure 5.17: Two examples using a predicted appearance model for unseen lighting categories. Figures are an unseen image (a), its expected appearance (b), and changes detected (in black) for 20° (c) and 40° (d) gaps in trained lighting. Change detection for the 20° predicted appearances is highly accurate, however for the low lighting angle in the bottom example the change detection for 40° has some false alarms from shadows which do not appear in either neighboring appearance model.

was repeated with skipping 3 lighting angles and predicting the middle.

Figure 5.17 shows some samples from this experiment. The renderings of the expected images are quite similar to the actual unseen images except in some shadow regions which are a mixture of their neighboring appearances shadow and non-shadow intensities. The change detection results are highly accurate as well, except when doing a wider prediction with low lighting angles, where shadows appear in the new image in places that are not in shadow for either neighboring appearance. These results suggest that predicting appearance in general will be possible except for regions in shadow, which are inherently unpredictable without a physical model for how shadows are cast.

5.4 Conclusions

In this section the core VAM framework has been extended to handle satellite imagery. The issue of variable illumination has been addressed with a local lighting appearance model which uses separate mixtures of Gaussians for groups of images with similar lighting directions. Haze and brightness effects are adjusted in a preprocessing step which uses the learned 3-d geometry and appearance to calculate an optimal gain and offset for each new

image. The modified VAM framework was shown to detect interesting structural changes in several neighborhoods of Baghdad with few false detections.

The ability to automatically detect changes in satellite imagery is a major advance that has the potential to revolutionize how intelligence data is used. Many terabytes of satellite imagery are collected by intelligence agencies every day, so much that they cannot possibly be fully analysed for interesting changes. Using VAM on a large region of interest one can obtain highly accurate change detection results in satellite images without any manual work on the part of image analysts. With this automated framework more important changes will be detected and fewer will be missed.

Chapter 6

Using VAM with Arbitrary Data Sources

The VAM framework presented in chapter 3 was designed for integration of aerial imagery from multiple sources. Using this framework, images from satellites can be used together with images from UAVs so long as the imaging devices are radiometrically consistent. However suppose that a road network of the region is available. There is no mechanism yet for using this data in the VAM framework, however a map projection isn't too different from standard aerial imagery and it shouldn't be difficult to incorporate its information into the model. In application there may be many potential sources of information about a region which are not standard images. Some examples relevant to aerial surveillance include LIDAR (light detection and ranging) data, SAR (synthetic aperture radar), and even man-made maps of the region. The goal of this section is to abstract the online updating scheme from chapter 3 to handle arbitrary sources of data.

When using data of multiple types together, one question to ask is what kind of information can be compared. For example, pixel intensity information from a greyscale image of a scene cannot be directly compared to an LIDAR elevation map image of the same scene, as the two data types have no correlation. However color images can be compared to greyscale images, so long as the equations to convert from color and greyscale are known. In general, a different appearance model will be needed for each type of image

data, and conversion equations may or may not exist for comparing appearance data. However the concept of voxel occupancy is relevant across all data types. The learned 3-d structure of the scene is the common base that can be used and updated by any appearance model. This allows different types of appearance data to improve each others estimates, even when the appearance types are incompatible.

This chapter has two sections. The first introduces a generalized version of the updating procedure from chapter 3. The second applies this generalized theory to the case of LIDAR. Update equations are derived in this section but no experiments are done.

6.1 General Updating

Suppose there are two sources of information about the world I_1 and I_2 and that information from I_1 has already been incorporated into the model (in previous chapters I_1 and I_2 were images). To integrate information from I_2 into the world two steps must be taken. First if I_1 and I_2 are the same type of appearance data then the appearance model of I_1 must be updated with I_2. If I_2 has appearance information but is incompatible with I_1 then a new appearance model must be initialized with I_2. Secondly the occupancy probabilities of all voxels must be updated. To do this the occupancy update equation 3.2.1 used to train the voxel world on a greyscale image can be generalized to use the arbitrary data in I_2. Then the occupancy probability of a voxel given the two pieces of information can be written as:

$$P(X \in S|I_1, I_2) = P(X \in S|I_1)\frac{P(I_2|X \in S, I_1)}{P(I_2|I_1)} \tag{6.1.1}$$

This gives the occupancy after observing both I_1 and I_2 as a multiplier times the occupancy after observing just I_1. This generalizes to n sources of information in the case that I_1 is all the information in the first $n-1$ sources. The multiplier here can be expanded several ways. In the case that I_2 is some kind of image (which includes regular greyscale/color/infrared images, DEMs, some forms of LIDAR, SAR, etc), it can be expanded along rays by conditioning on the visible voxel as in equation 3.2.2:

$$\frac{\sum_{X' \in R} P(I_2|V = X', I_1)P(V = X'|X \in S, I_1)}{\sum_{X' \in R} P(I_2|V = X', I_1)P(V = X'|I_1)} \tag{6.1.2}$$

where the $P(V = X'|I_1)$ are computed as before and the $P(I_2|V = X', I_1)$ are computed differently depending on the data type of I_2. If I_2 has appearance information about the world surface then this probability will be computed using an appearance model for that data type, most likely a mixture of Gaussians or something similar. If I_2 has information about the 3-d structure of the scene then there will be no appearance model and the probability will be computed using the current occupancy probabilities and knowledge about imaging process. An example of this type of update for LIDAR imagery is given in the next section.

It may be however that I_2 is not formed by any kind of imaging process. For example it could be a man-made map, an externally generated 3-d model of the scene, or point cloud LIDAR. In this case the expansion of equation 6.1.1 along camera rays no longer makes sense since there are no cameras. However equation 6.1.1 can still be broken down further as:

$$P(X \in S|I_1, I_2) = P(X \in S|I_1)\frac{P(I_2|X \in S, I_1)}{P(I_2|X \in S, I_1)P(X \in S|I_1) + P(I_2|X \notin S, I_1)P(X \notin S|I_1)} \tag{6.1.3}$$

which gives the update in terms of $P(I_2|X \in S, I_1)$ and $P(I_2|X \notin S, I_1)$ which will be data type dependent. For example in the case of an external 3-d model $P(I_2|X \in S, I_1)$ might be high when there is 3-d geometry in I_2 near X and low when not.

Equations 6.1.2 and 6.1.3 are abstract update equations for using arbitrary data types to update the estimates of voxel occupancy probability. When extending the VAM framework to work with new data types these equations will be the starting point.

6.2 LIDAR Updating

In this section the general updating theory is applied to the special case of LIDAR data. LIDAR is an optical remote sensing technology that estimates distance to the scene by

Figure 6.1: Two common forms of LIDAR are the raw point cloud output (left) and a processed digital elevation map (DEM) where brighter pixel intensities indicate taller structures (right). Left image copyright Ambercore Software, Inc.

measuring properties of scattered laser light. A LIDAR dataset contains 3-d range information about the scene, and typically comes in one of two different forms (see figure 6.1). The first is the raw 3-d point cloud of the returns from the LIDAR collector. The second more processed form is a digital elevation map (DEM) of the scene, an image looking top down on the world with the height of the world at each pixel indicated by intensity.

It is important to note that LIDAR in both data formats gives incomplete information about the 3-d structure of the scene. Aerial LIDAR is taken with a laser that sweeps along the ground as the plane flies overhead, and as such it will have few returns on surfaces tangent to the sweeping plane, such as sides of buildings. It may also have missing returns on occluded objects if the flight plan doesn't have adequate coverage. In addition there are measurement errors in the LIDAR returns, which result in the 3-d data being shifted some distance from its true 3-d location. It will be assumed here that this shift can be modeled as a normal random variable with mean 0 and some variance determined by the collection system. It is also assumed in everything that follows here that the LIDAR data is perfectly registered with the voxel world, so that the only errors in 3-d alignment are these measurement errors.

In the case of DEM LIDAR there may also be the concept of a first and second return, where the first return is the strongest laser response and the second return is the second peak in the response. For most surfaces the two responses will be the same, however for vegetation and edges of buildings there may be multiple distinct returns. Unfortunately the VAM framework cannot interpret the second return because there is no concept of a partially occluding surface such as a tree. The options are to either ignore the second return and update using just the first, or to not do any updating when the first and second returns are significantly different. The later is probably the better choice, since in the case of vegetation the first return may actually be somewhere in the middle of the tree and updating voxels there will decrease the probability of voxels on the outer tree surface.

6.2.1 DEM LIDAR

LIDAR given in elevation map format can be used to update the voxel world using equation 6.1.2. In this case the rays from the elevation map image are vertical rays in the world, and the LIDAR indicates where the highest surface on each ray lies. Consider one such ray R which intersects the LIDAR image at pixel x, y where the height is $L_{x,y}$. The question is what the probability $P(I_2|V = X, I_1)$ in equation 6.1.2 should be in this case when I_2 is the LIDAR data L. If the LIDAR was a perfect measurement then it would make sense for $P(L|V = X, I_1) = P(L|V = X)$ to be uniformly distributed over the voxel V and 0 outside. However as mentioned before the 3-d LIDAR data will be some unknown normally-distributed translation D away from the actual 3-d surface. To compute the LIDAR probability $P(L|V = X)$ exactly one needs to integrate:

$$\int P(L|V = X, D)P(D)dD = \tag{6.2.1}$$

$$\int\int P(L|V = X, D)P(D_z)dD_z P(D_{xy})dD_{xy} \tag{6.2.2}$$

where the independent height and translation components of the error direction D are separated and integrated separately. Consider first the inner integration against D_z when D_{xy} is fixed. With D_{xy} fixed the ray R passes through pixel $x + D_x, y + D_y$ where the height is $L_{x+D_x,y+D_y}$ or $L_{D_{xy}}$ for short. Then the inner integral in equation 6.2.2 is:

$$\int P(L_{D_{xy}}|V = X, D_z)P(D_z)dD_z \tag{6.2.3}$$

which can easily be computed since $P(D_z)$ is normally distributed with mean 0 and fixed variance σ^2 and the height probability $P(L_{D_{xy}}|V = X, D_z)$ is by previous remarks uniformly distributed on $[X_b + D_z, X_t + D_z]$ where X_t and X_b are the top and bottom z-coordinates of voxel X.

To compute the outer integral in equation 6.2.2 note that the inner integral is the same for all D_{xy} that map into the same LIDAR image pixel. The outer integral then becomes a sum over all LIDAR pixels in a neighborhood of x, y:

$$\sum_{LP \in N(x,y)} \int P(L|V = X, D_z, LP)P(D_z)dD_z P(LP) \tag{6.2.4}$$

Since the D_{xy} were normally distributed this will be the product of a Gaussian mask and the inner integrals computed at neighboring pixel locations. The result is that voxels near the LIDAR heights will have high probability and those far away will not, where "near" can be tuned by setting the height and translation variances of the error direction D appropriately. This updating of the world with DEM LIDAR is very similar to normal grayscale image updating and can be done with almost the same code.

6.2.2 Point-Cloud LIDAR

Suppose instead now that the LIDAR is given as a set of unorganized 3-d points, as is typical of raw LIDAR data. In this case there is no concept of an image, so equation 6.1.3 must be used to update the occupancy probabilities. This situation is much harder to model than the DEM case since one has to take into account the LIDAR point spacing and the possibility of surfaces with no observed LIDAR points, such as sides of buildings (not an issue with DEM LIDAR since the visibility probability allows occlusion by the roof). It is also difficult to talk about $P(L)$ in this case since the L is an unordered set of 3-d points of variable size.

To make the problem tractable it will be assumed that for each voxel X the distance D_X to the closest LIDAR point is a sufficient statistic for estimating the occupancy probability

at X. That is, the distance D_X contains all relevant information from the full LIDAR point cloud L about the occupancy of voxel X. As such the terms $P(I_2|X \in S, I_1)$ and $P(I_2|X \notin S, I_1)$ in update equation 6.1.3 become $P(D_X|X \in S)$ and $P(D_X|X \notin S)$. Once one knows how to calculate these probabilities one can update the voxel occupancy probabilities.

Consider first $P(D_X|X \in S)$, the density of the distance D_X to the nearest LIDAR point when X is an actual surface voxel. One might be tempted to model this with a chi distribution (if a vector distance D is normally distributed with mean 0 then its norm $||D||$ has a chi distribution). However it is necessary to factor in the possibility that the voxel lies on a surface yet there are no LIDAR point observations anywhere near it. This occurs frequently especially on surfaces parallel to the LIDAR scanning plane like sides of buildings. Let $X \in L$ denote the state of X belonging to a LIDAR observed surface (note this is stronger than $X \in S$). Then one can expand $P(D_X|X \in S)$ as:

$$P(D_X|X \in L)P(X \in L|X \in S) + P(D_X|X \notin L, X \in S)P(X \notin L|X \in S) \tag{6.2.5}$$

Here $P(D_X|X \in L)$ (P_D for short) can be assumed to be the afforementioned χ distribution. The probability $P(X \in L|X \in S)$ of a surface producing a LIDAR point is a fixed constant p_{ls} to be determined experimentally. In a typical collection around 80-95% of the surfaces are not observed by the LIDAR device and so a p_{ls} of .8-.95 would be appropriate. The probability density $P(D_X|X \notin L, X \in S)$ is a bit harder to quantify. This is the probability of the distance from X to the nearest LIDAR point when X is not observed by the LIDAR. This distance can be anything from 0-∞ since there is no information about how close the nearest LIDAR surface might be. It will be assumed that this distance is uniformly distributed on a large but bounded range so that $P(D_X|X \notin L, X \in S) = k$ for some small constant k. Putting these together:

$$P(D_X|X \in S) = P_D p_{ls} + k(1 - p_{ls}) \tag{6.2.6}$$

One can argue that also $P(D_X|X \notin S) = k$ since when X is not a surface voxel the

distance to the nearest LIDAR point is again arbitrary. Plugging these terms into the update multiplier in equation 6.1.3 and simplifying one obtains:

$$P(X \in S|I_1)\frac{p_{ls}P_D + (1-p_{ls})k}{(p_{ls}P(X \in S|I_1))P_D + (1-p_{ls}P(X \in S|I_1))k} \tag{6.2.7}$$

This update scheme behaves as one would like, either decreasing or increasing depending on whether k or P_D dominates the above multiplier. When X is near a LIDAR point P_D will be high and $P(X \in S)$ will become near 1 after the update. If X is far enough away from the LIDAR however, k while small will be greater than P_D and $P(X \in S)$ will decrease by an amount determined by the parameter p_{ls} and the current estimate of $P(X \in S)$.

6.3 Conclusions

In this section VAM has been abstracted to allow its use with arbitrary data types. Voxel occupancy update equations are given in the cases where the data type is an image and when it is not. To demonstrate how these equations could be used with non-appearance data, examples for both image and point-cloud LIDAR are sketched out.

This gives just a hint at how extensible the VAM framework truely is. With the theory in this section, VAM can now use any type of information available about the scene to make decisions about what has changed. This is especially useful in surveilance applications, since often many different types of data are available for a scene, and VAM can seamlessly integrate in whatever data is available as it is collected. The key here is the voxel occupancy probabilities, which are the "glue" that allow incompatible data types to communicate with each other.

Chapter 7

Implementation

The VAM algorithm described in previous chapters provides a comprehensive solution to the 3-d change detection problem. However the theory is only half of the solution, as the algorithm must be implemented efficiently if it is to ever be used in any real system. Designing an efficient storage system for a large array of voxels is particularly important as improperly managing gigabytes of voxel memory can result in severe slowdowns.

Even for a small voxel volume, the VAM algorithm presented in chapter 3 requires more storage for voxel data than can be easily stored in physical memory. With 3-mode mixture of Gaussians appearance model and occupancy probability, each voxel requires 10 floats or 40 bytes per voxel of memory. For a typical $1000 \times 1000 \times 100$ voxel scene of a city block the model will require 4GB of memory and will be much greater if a larger region or lighting dependent appearance model from chapter 5 is needed.

It is clear that the voxel data must be stored on disk, and as reading and writing to disk will be a bottleneck in performance, an appropriate data structure must be used to make traversing the voxel array as efficient as possible. Two frameworks were used during the development of the algorithm, one storing voxels in a stack of planes perpendicular to the z-axis and one storing voxels in larger "supervoxel" blocks. The advantages and disadvantages of both schemes are described in the next two sections. Answers to many other questions that arise during implementation are given in the last section.

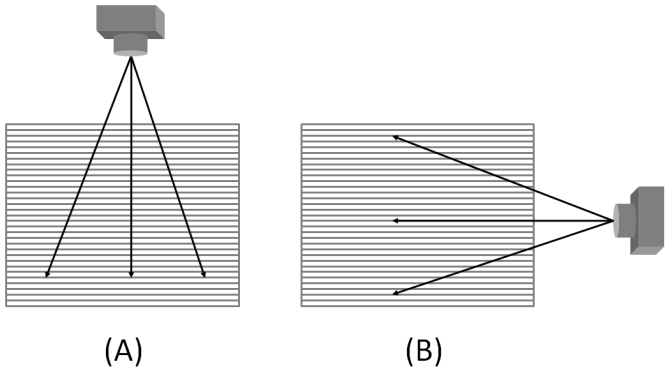

Figure 7.1: Camera rays intersecting a voxel volume organized in planes parallel to the z-axis. In (A) the camera position is such that all rays intersect the planes in top-down order, making occlusion calculations easy. In (B) with the side camera, rays above the middle ray intersect the stack in one order while rays below intersect in the reverse order. This complication, in addition to the need to pay close attention to the order of voxels in each plane, makes occlusion computations harder to implement in a plane stack framework.

7.1 Plane Stack

Storing voxels in sheets of planes parallel to the z-axis is the most efficient framework for VAM when all images taken are from top-down viewpoints (see figure 7.1A). Under this assumption, rays from the camera pass through the stack of planes from top to bottom, and occlusion probabilities can be computed quickly by reading, processing, and writing back each plane in order. Efficient implementations of the updating and change detection algorithms are described next.

Suppose a VAM model is stored in planes $\{l_z\}_{z=0}^{Z}$ and that one wishes to update the model on a new image I with camera matrix C. Assume that the camera is far from the world surface and the voxel spacing is such that for each pixel in the image the backprojected ray of voxels is roughly contiguous (if not see section 7.3.1). Pseudocode for the update procedure described next is given in Algorithm 1. This pseudocode is written in an object-oriented way where $\rightarrow$ denotes a member function on a class. For example $appearance(vx, vy, vz) \rightarrow update(color)$ tells the *appearance* object at coordinate vx, vy, vz to update itself using the given *color* observation.

Two passes through the planes are necessary to train the voxel model on the new image I. The first pass updates the appearance model in each voxel and computes $P(I_y)$

Algorithm 1 Plane slice updating of a $VX \times VY \times VZ$ voxel volume with a $IX \times IY$ image I with camera matrix C.

```
// Fill the following IX × IY matrices:
unoccluded(ix, iy) = 1.0
py_full(ix, iy) = 0.0

// First pass
for vz = VZ to 1 do
  load appearance(vz)
  load occupancies(vz)
  for vx = 1 to VX do
    for vy = 1 to VY do
      voxel_center = world_coords(vx, vy, vz)
      voxel_proj = C×voxel_center
      voxel_color = I( voxel_proj )
      appearance_prob = appearance(vx, vy, vz) →probability( voxel_color )
      appearance_weight = occupancies(vx, vy, vz)×unoccluded( voxel_proj )
      appearance(vx, vy, vz) →update( voxel_color, appearance_weight )
      py_partial(vx, vy, vz) = py_full( voxel_proj ) +
        unoccluded( voxel_proj )×appearance_prob
      py_full( voxel_proj ) += occupancies(vx, vy, vz)×
        unoccluded( voxel_proj )×appearance_prob
      unoccluded( voxel_proj ) *= ( 1 - occupancies(vx, vy, vz) )
      projections(vx, vy, vz) = voxel_proj
    end for
  end for
  save appearance(z)
  save projections(z)
  save py_partial(z)
end for

// Second pass
for vz = VZ to 1 do
  load occupancies(vz)
  load projections(vz)
  load py_partial(vz)
  for vx = 1 to VX do
    for vy = 1 to VY do
      voxel_proj = projections(vx, vy, vz)
      occupancies(vx, vy, vz) *= py_partial(vx, vy, vz) / py_full(voxel_proj)
    end for
  end for
  save occupancies(z)
  delete projections(z)
  delete py_partial(z)
end for
```

for every pixel y in the image. The second pass updates the occupancy probability for every voxel. Note that the update cannot be done in a single pass because to compute $P(I_y)$ for a pixel, one needs to access every voxel on its ray and each voxel on the ray needs $P(I_y)$ to be precomputed before it's occupancy probability can be updated.

The first pass is done primarily to compute $P(I_y)$, however it is convenient to update the appearance models during this pass and to save some computed quantities temporarily to disk so that they don't need to be recomputed during the second pass. During this pass, each z-slice of voxels is processed as follows. First, the occupancy probabilities and appearance models for every voxel in the plane are loaded into physical memory from disk. Then each voxel X in this slice is projected into the image via the known camera matrix C and assigned to the unique pixel y which is closest to $C(X)$. For each pixel, running tallies of occlusion and color probabilities are kept and updated as more planes are passed through. The occlusion tally at pixel y is used to compute the visibility probability $P(V = X)$ at voxel X which then is used to update the appearance model at X. The color probability tally is increased by $P(I_y|V = X)P(V = X)$ so that the quantity $P(I_y)$ has been computed after the last plane has been processed.

Quantities that can be saved to disk during this pass to reduce repeated computation are the voxel projection coordinates $C(X)$ and the conditional color probability $P(I_y|X \in S)$. Saving the projections will be a speed up as long as the time required to read and write the coordinates from disk is less than the time required to reproject all the voxels into the image. Saving the conditional color probability however will always be more efficient since it only takes up 1 float-per-voxel and it eliminates the need to reload the 9+ float-per-voxel appearance model on the second pass.

If these quantities are computed and saved then very little needs to be done on the second pass. For each plane the occupancy probabilities for each voxel are multiplied by the ratio $P(I_y|X \in S)/P(I_y)$, both terms of which have been pre-computed and stored. After updating the occupancy probabilities for each plane the extra saved quantities can be deleted from disk.

Detecting changes in a new image can be implemented in a similar way, though only one pass is required since no state variables need to be updated. Pseudocode for this

process is given in algorithm 2.

Algorithm 2 Plane slice change detection of a $VX \times VY \times VZ$ voxel volume with a $IX \times IY$ image I with camera matrix C.

```
// Fill the following IX × IY matrices:
unoccluded(ix, iy) = 1.0
py(ix, iy) = 0.0

for vz = VZ to 1 do
  load appearance(vz)
  load occupancies(vz)
  for vx = 1 to VX do
    for vy = 1 to VY do
      voxel_center = world_coords(vx, vy, vz)
      voxel_proj = C×voxel_center
      voxel_color = I( voxel_proj )
      appearance_prob = appearance(vx, vy, vz) →probability( voxel_color )
      py( voxel_proj ) += occupancies(vx, vy, vz)×
        unoccluded( voxel_proj )×appearance_prob
      unoccluded( voxel_proj ) *= ( 1 - occupancies(vx, vy, vz) )
    end for
  end for
end for
return py
```

Though each plane must be fully processed before the one below it can be processed, there are many opportunities for parallelization during the processing of a single plane. Once a plane is loaded into memory the updating or appearance probability calculation of each voxel can be performed independently by separate processors. In a system with multiple hard drives the time spend loading and saving planes can be reduced by storing adjacent planes on different disks, so that a plane can be pre-loaded into memory while the plane above it is still being processed. Finally, it is also possible to calculate the projection of each voxel into an image using common graphics hardware, which will greatly reduce the running time of this step.

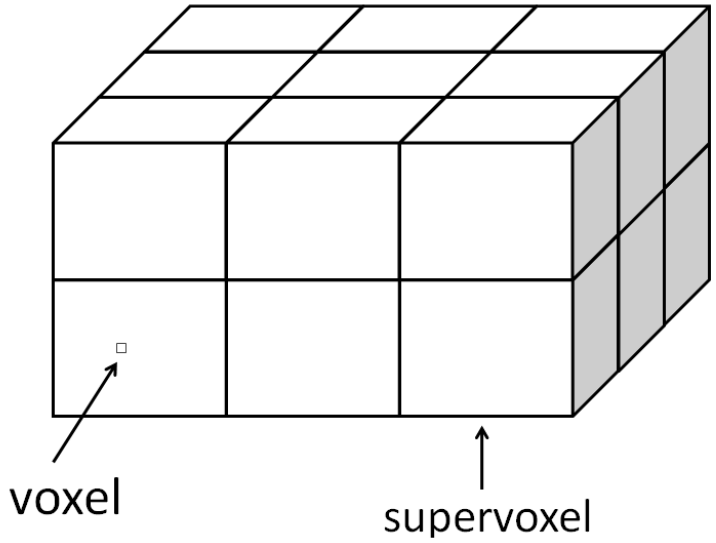

Figure 7.2: Voxels can be grouped into larger "supervoxel" blocks which are loaded from and saved to disk together.

7.2 Supervoxels

Storing voxels in planes is appropriate when it can be guaranteed that all rays from the camera pass through the z-planes in the same order. In many ground surveillance applications this may not be possible. When the camera is close to the ground, rays can intersect the planes in different order (see figure 7.1B). In addition rays will often intersect many voxels in each plane making the order of traversal of each plane important. These obstacles can be overcome with the voxel-plane approach, however it will be easier to store voxels is in larger "supervoxel" blocks (see figure 7.2). This scheme also has the advantage of making the update and change detection algorithms more natural to implement, since whole rays of voxels are loaded into memory at the same time.

The updating and change detection algorithms will be implemented differently when using supervoxels. To train the volume on a new image, the image is first subdivided into sub-blocks of appropriate size (50 voxels per edge was used in experiments). For each sub-image, all supervoxels containing any voxels which project into it are loaded from disk into physical memory. For each pixel in this sub-image the ray of voxels projecting into it are identified and updated using the equations in chapter 3. After processing all pixels in

this sub-image, all loaded supervoxels are saved back to disk. Note that this process can be done for any arbitrary camera position and orientation, since all necessary voxels are pre-loaded into memory. The implementation of the change detection algorithm is similar.

This is a fairly efficient scheme. The main source of computational inefficiency is loading more voxels than necessary into memory for each sub-image. When the supervoxel blocks are small this will be less of a problem, however large supervoxels benefit from being able to read large contiguous blocks of memory from disk, so there is a trade off. A supervoxel implementation of VAM was used for the majority of experiments in this thesis. A newer version using a plane stack was implemented later on and runs on average 2-5 times as fast, though this may be partially the result of smarter memory management.

7.3 Additional Implementation Issues

The plane stack and supervoxel implementations of VAM presented in the previous sections are the core of an efficient implementation of the VAM. There are still a handful of unanswered implementation questions and miscellaneous speedups that are addressed in this section.

7.3.1 Voxel/Image Resolution

The resolution of the voxel grid should be decided based on speed and storage requirements as well as the size of the smallest change the system needs to be able to detect. After the voxel resolution is set, the resolution of every training image must be adjusted to line up well with that resolution before it can be processed. This image resizing can be done with any standard interpolation method including bilinear or bicubic interpolation and by post-multipling the camera matrix with an equivalent scaling matrix.

The reason that this rescaling is necessary is that rays of voxels projecting into a pixel in the image should ideally have unit thickness (see figure 7.3). If the rays are too many voxels thick or have gaps from missing voxels then the occlusion calculations may be inaccurate. For example if a ray has too many gaps and is missing a true surface voxel, then all voxels below will potentially have their appearance models corrupted during

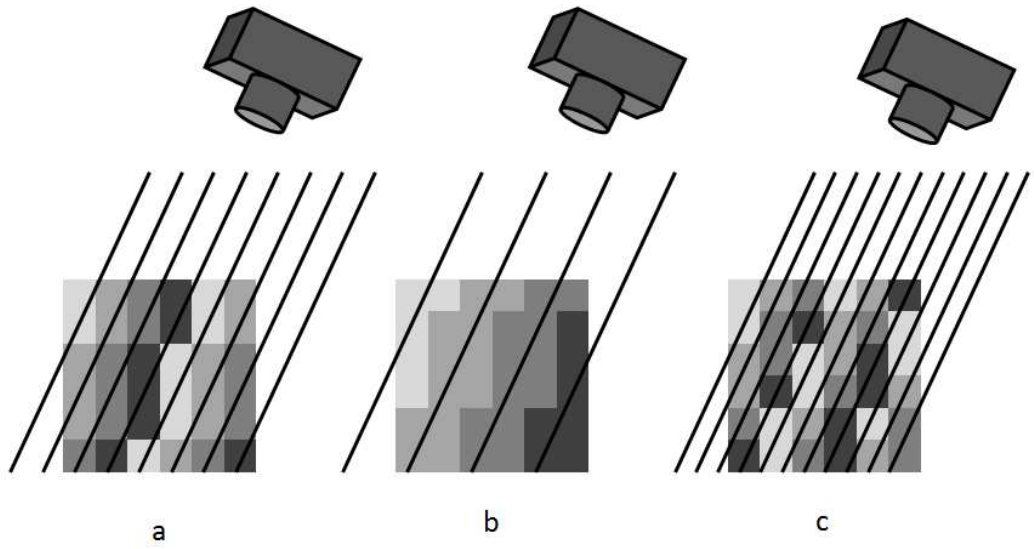

Figure 7.3: Assignment of voxels to camera rays for different image resolutions. Camera rays are shown as black lines while voxels are shaded grey according to which ray they belong. The resulting voxel-rays are a) of unit thickness when pixel spacing is just right, b) too thick when pixel spacing is too large, and c) discontinuous when pixel spacing is too small.

learning when the mismatched pixel intensity is not occluded. Of course, unit thickness is hard to guarantee in practice but good results will be obtained as long as all rays have near unit thickness.

Considerable care must be taken when choosing the rescaling of each image so that the resulting voxel rays have appropriate thickness. Figure 7.3 shows the consequences of having inappropriate pixel-scaling. One scheme for automatically calculating an appropriate scaling is to pick a voxel, project it and its 26 voxel neighbors into the image, and take the greatest image-distance to a neighbor as the scaling factor. A more detailed explanation is given in algorithm 3.

This discussion assumes that all images taken of the scene will have the same resolution, or that no relevant changes will be missed if all images are scaled down to the coarsest resolution. No work has been done to handle the case where changes need to be detected for a variety of different resolutions, though it should be an easy extension of the algorithms

Algorithm 3 Determine the scaling factor s for an image with camera C for a voxel volume of voxel spacing dv.

Calculate the principal ray r of the camera.
Find the voxel V at the intersection of r and the top of the volume.
$V_i = C \cdot V$
$D = 0.0$
for $dx = -dv, 0, dv$ **do**
  for $dy = -dv, 0, dv$ **do**
    for $dz = -dv, 0, dv$ **do**
      $W = V + (dx, dy, dz)$
      $W_i = C \cdot W$
      $d = \|V_i - W_i\|$
      $D = \max(D, d)$
    end for
  end for
end for
$s = 1/D$

presented here. For example one could store multi-resolution voxel volumes in an oct-tree, and use an appropriately sized voxel grid for each image, with voxel grid dependency going from coarse to fine, but not the reverse.

7.3.2 Normalization Implementation

To do the normalization suggested in chapter 5, it is necessary to the compute the probability of an image $P(I)$ quickly and efficiently for many different images I which have different gains and offsets. To compute $P(I)$ for just one image one needs to first compute the probability of each pixel, which requires a full traversal of the voxel volume. This straightforward approach to calculating $P(I)$ will be too slow since a full voxel volume traversal is needed for each image.

Since the camera of the adjusted image never changes, it is much more efficient to first transport the appearance information in the voxels to the 2-d image domain along camera rays. To do this one constructs a new image which contains a mixture of Gaussians at each pixel. Each mixture is trained using the mixture models in each voxel along its ray in the camera. Though there is undoubtedly an optimal technique for merging N mixtures of Gaussians into 1, it has proved fast and convenient to simply train the new mixture

on the means from all the modes in all the voxels along the ray. For a mode with weight w in voxel X, the new mixture of Gaussians was trained on the mean of that mode with weight $wP(V = X)$ to insure that only modes with high weight and visibility probability contribute to the new mixture.

The resulting mixture of Gaussians image can be used to calculate $P(I)$ approximately with only a 2-d image traversal rather than a full 3-d voxel traversal. This makes a brute-force search of normalization parameters computationally feasible. This "probability image" is potentially useful for more than just normalization calculation, for instance it could be used to compute image offset parameters for registering a new image.

7.3.3 Miscellaneous Speedups

Over the years of development a number of other speed optimizations were also discovered. A few of the more significant implementation tips are listed below.

- After training on enough images, voxels with low enough occupancy probability can be ignored in future computations. Specifically an occupancy probability of 0 can be used, and no appearance updating or probability calculation need be done, which are two of the most expensive voxel operations. These ignored voxels can be brought back into play if they ever project into a pixel which has low probability, so that the algorithm is still able to adapt to permanent changes in the 3-d scene.

- Enforce minimum/maximum bounds on many of the online variables such as occupancy probability and variances of mixture modes. Unless using the voxel-removal technique just described, the occupancy probabilities should never reach 0.0 or 1.0 because once at these values they cannot be changed by the update multiplier.

- All variables that need to be stored to disk (occupancy, Gaussian mixture parameters, etc.) are bounded floating point numbers. However given the statistical uncertainty of these variables, 8 bits is enough to represent any parameter with the needed accuracy. Storing these variables as bytes on disk and converting to floats when loaded reduces the disk access time by a factor of 4.

7.4 bvxm

At the date of publication of this thesis, an implementation of the VAM algorithm is available as part of the free VXL computer vision libraries in the brown contrib area at **contrib/brl/bseg/bvxm**. This library is implemented using the plane stack framework described in this chapter. As new uses for VAM have been discovered additional functionalities have been added to **bvxm** beyond what has been described in this thesis (hence **bvXm** rather than **bvAm**).

Chapter 8

Conclusions

This thesis has presented the first truly comprehensive solution to the 3-d change detection problem. Previous work has only provided partial solutions which either limit the amount of allowed camera motion or detect only certain types of change. These attempts fall short because they don't attempt to solve the difficult but necessary problem of 3-d reconstruction which allows surfaces to be compared across all images.

The VAM framework solves this problem by using a voxel volume to model both surface occupancy and appearance. The model can be updated online with each new image using the Bayesian update procedure presented in chapter 3. Changes are easily detected as those pixels who when backprojected onto the learned 3-d world have low appearance probability. This basic framework can be extended to handle satellite images which are taken under different outdoor lighting conditions and contain haze.

This approach has been demonstrated to work well on real images, producing accurate results on any data set thrown at it. These include images taken by helicopter of challenging urban scenes with low viewing angles, complicated buildings, and scattered vegetation. Even more exciting are the kinds of interesting structural changes detected in satellite images taken over the span of years. The accuracy and robustness of VAM on these challenging real world data sets shows how powerful a framework it is.

Looking back there are a few important insights in regard to why VAM works so well. The first is that the use of voxels, while novel in this context, is actually necessary for a 3-d change detection algorithm to work online. Only with a full volumetric representation

can hypotheses for all possible surfaces be maintained throughout time. If one doesn't use a volumetric model and a new structure appears in the scene, the algorithm will have to refer back to previous images to learn the new structure, making the algorithm offline.

Another key component to VAM's success is how it avoids making any decisions until they are absolutely needed. Rather than reconstruct an explicit 3-d surface as some intermediate step and use it to detect changes, VAM carries the uncertainties of the world surface along all the way into the final change detection calculation. By doing this no potentially important information about what the correct surface position may be is lost. Only when the final change/no-change map is needed are all the probabilities summed up. This idea of keeping uncertainties around as long as possible may be useful in many computer vision problems.

There is still plenty of work left to be done to improve upon VAM, starting with more sophisticated appearance models. The local lighting model presented in chapter 5 is a logical first attempt at handling images with different illumination conditions, however there may be better alternatives which give more accurate detections while requiring fewer training images. One possibility is building a higher level BRDF model on top of the existing local model so that new lighting directions can be predicted, while keeping the local model to learn shadows.

Another high priority task is developing an offline version of the VAM algorithm for when a batch of images are available all at once. One of the key advantages of VAM is the ability to update the background voxel model online as each new image is taken. However when many images are available at once, one should be able to achieve a more accurate reconstruction by considering all of the images together rather than just one image at a time. One can convert the online VAM algorithm into an offline algorithm by simply iterating over the same set of images many times, however there are undoubtedly better and faster ways of doing this.

Speed is another issue that hasn't been a huge priority in the development of the VAM framework, yet to be usable in a real-world system a much faster implementation will be needed. This will be no small undertaking, involving a detailed analysis of running time to identify and remove all bottlenecks. However it is safe to say that the biggest speed

gains will be obtained through parallelization either with multi-core processors or with programmable graphics cards (NVIDIA's CUDA framework for example).

Many other improvements to VAM have been made by other graduate student researchers in the Brown LEMS lab. The issue of using images with different resolutions in the same VAM framework has been solved by maintaining a pyramid of linked voxel grids (similar to an oct-tree) from which an appropriate grid is chosen for each image based on its resolution. The mixture of Gaussians appearance model has been modified to work with grayscale, RGB, and multi-spectral image types.

Other graduate student researchers have also solved the problem of automatic camera correction. The VAM algorithm requires that the input images are associated with camera models accurate to within a pixel of reprojection error. In application it is rare for computed camera models to have this level of accuracy, so it some form of automated camera correction will be necessary before images can be input into the VAM algorithm. Such automated correction algorithms have been developed in the Brown LEMS lab which make use of the existing VAM framework. For correcting 3x4 camera matrices in aerial video, a technique has been developed which finds the camera adjustment that maximizes the image probability with respect to a current estimate of the voxel world. For correcting the rational polynomial cameras in satellite images a similar technique has been developed which uses image edge statistics rather than pixel intensities to compute the optimal image offset.

These are just some of the potential avenues for future work to improve change detection results with VAM. But perhaps the most exciting possibilities are for using VAM for more than just change detection. After training VAM on a set of images, one obtains a reasonably accurate 3-d reconstruction of the scene infused with appearance information (possibly from multiple sources), which can be easily updated at a later time when more information is available. Moreover this object is able to draw itself and compute the probabilities of images taken of it.

Such an object would certainly be useful for many computer vision problems besides change detection. A few non-change detection computer vision projects have begun at the Brown LEMS lab which use VAM as a basis. Some effort has been made to clean up

the learned voxel geometry so that the recovered surfaces could be used for visualization purposes or to create DEM height maps. There have even been some initial experiments using VAM for object recognition where a test image is compared to several learned voxel models to determine its most probable class. After four years of research, the task of exploring all of the possibilities for VAM has just begun.

Bibliography

[1] T. Aach, A. Kaup, and R. Mester. Statistical model-based change detection in moving video. *Signal Processing*, 31, 1993.

[2] Motilal Agrawal and Larry S. Davis. A probabilistic framework for surface reconstruction from multiple images. In *CVPR*, 2001.

[3] M. J. Black, D. J. Fleet, and Y. Yacoob. Robustly estimating changes in image appearance. *Computer Vision and Image Understanding*, 78, 2000.

[4] Mohamed Borchani, Florence Cloppet, Volkan Atalay, and Georges Stamon. Change detection in aerial images. In *First Canadian Conference on Computer and Robot Vision*, 2004.

[5] Adrian Broadhurst, Tom Drummond, and Robert Cipolla. A probabilistic framework for space carving. In *ICCV*, 2001.

[6] Mark J. Carlotto. Detection and analysis of change in remotely sensed imagery with application to wide area surveillance. *IEEE Trans on Image Processing*, 2(3), 2005.

[7] W. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *Vision algorithms theory and practice workshop*, 1999.

[8] Xiaolong Dai, Siamak Khorram, and H. Cheshire. Automated image registration for change detection from landsat thematic mapper imagery. In *IGARSS*, 1996.

[9] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelyhood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 1977.

[10] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[11] D. Farin and P.H.N. de With. Misregistration errors in change detection algorithms and how to avoid them. In *IEEE International Conference on Image Processing*, 2005.

[12] Oscar Firschein and Thomas M. Strat. *RADIUS: Image Understanding for Imagery Intelligence*. Morgan Kaufmann, 1997.

[13] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice*. Addison-Wesley, 2 edition, 1997.

[14] F. Del Frate, G. Schiavon, and C. Solimini. Use of high resolution satellite data for change detection in urban areas. In *ESA-EUSC*, 2005.

[15] Nir Friedman and Stuart Russell. Image segmentation in video sequences: a probabilistic approach. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

[16] A. F. Habib, R. I. Al-Ruzouq, and C. J. Kim. Semi-automatic registration and change detection using multi-source imagery with varying geometric and radiometric properties. In *ISPRS*, 2004.

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in compute vision*. Cambridge University Press, 2000.

[18] Richard I. Hartley and Tushar Saxena. The cubic rational polynomial camera model. In *DARPA Image Understanding Workshop*, 1997.

[19] Eric Hayman and Jan-Olof Eklundh. Statistical background subtraction for a mobile observer. In *ICCV*, 2003.

[20] Aaron J. Heller, Yvan G. Leclerc, and Quang-Tuan Luong. A framework for robust 3-d change detection. In *Int'l Symposium on Remote Sensing, SPIE*, 2001.

[21] Y. Z. Hsu, H. Nagel, and G. Reckers. New likelyhood test methods for change detection in image sequences. *Computer Vision, Graphics, and Image Processing*, 26, 1984.

[22] J. Hu, S. You, and U. Neumann. Approaches to large-scale urban modeling. *IEEE Compute Graphics and Applications*, 23(6), November 2003.

[23] Andres Huertas and Ramakant Nevatia. Detecting changes in aerial views of man-made structures. In *International Conference on Computer Vision*, 1998.

[24] S. Huwer and H. Niemann. Adaptive change detection for real-time surveillance applications. In *Visual Surveillance*, 2000.

[25] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[26] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multi-sensor video surveillance. In *DARPA Image Understanding Workshop*, volume 1, 1998.

[27] K. Kutulakos and S. Seitz. A theory of shape by space carving. *IJCV*, 2000.

[28] Weiming Li, Xiaoming Li, Yihong Wu, and Zhanyi Hu. A novel framework for urban change detection using vhr satellite images. In *ICPR*, 2006.

[29] Anurag Mittal and Dan Huttenlocher. Scene modeling for wide area surveillance and image synthesis. In *CVPR*, pages 160–167, 2000.

[30] S. G. Narasimhan and S. K. Nayar. Removing weather effects from monochrome images. In *CVPR*, 2001.

[31] Bui-Tuong Phong. Illumination for computer generated pictures. *CACM*, 18(6), June 1975.

[32] Thomas Pollard and Joseph L. Mundy. Change detection in a 3-d world. In *CVPR*, 2007.

[33] W. K. Pratt. *Digital Image Processing.* John Wiley and Sons, 1991.

[34] Richard J. Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image change detection algorithms: A systematic survey. *IEEE Trans on Image Processing, Vol. 14, No. 3*, 2005.

[35] Y. Ren, C. Chua, and Y. Ho. Statistical background modeling for non-stationary camera. *Pattern Recognition Letters*, 24, January 2003.

[36] A. L. Reno and D. M. Booth. Change detection in aerial image sequences. In *First Int'l Workshop on Image and Signal Processing and Analysis*, 2000.

[37] Christof Ridder, Olaf Munkelt, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *International conference on recent advances in mechatronics*, 1995.

[38] P. Rosin and E. Ioannidis. Evaluation of global image thresholding for change detection. *Pattern Recognition Letters*, 24(14), 2003.

[39] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 1999.

[40] P. Smits and A. Annoni. Toward specification-driven change detection. In *IEEE Trans. Geoscience and Remote Sensing*, 2000.

[41] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, volume 2, pages 246–252, 1999.

[42] Hai Tao, Harpreet Sawhney, and Rakesh Kumar. Dynamic layer representation with applications to tracking. In *CVPR*, 2000.

[43] David Williams. *Probability with Martingales.* Cambridge University Press, 1991.

[44] Hulya Yalcin, Robert Collins, Martial Hebert, and Michael J. Black. A flow-based approach to vehicle detection and background mosaicking in airborne video. In *Video Proceedings in conjunction with CVPR'05*, June 2005.

[45] Annie Yao and Andrew Calway. Dense 3-d structure from image sequences using probabilistic depth carving. In *ECCV*, 2004.