Perceptual Data Mining: Bootstrapping visual intelligence from tracking behavior

by

Christopher P. Stauffer

B.S. Computer Science, Northwestern University, 1995
B.S. Electrical Engineering, Northwestern University, 1995
B.S. Biomedical Engineering, Northwestern University, 1995
M.S. Electrical Engineering and Computer Science, M.I.T., 1997

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2002

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2002

Certified by.....
W.E.L. Grimson
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by..
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Perceptual Data Mining: Bootstrapping visual intelligence from tracking behavior

by

Christopher P. Stauffer

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2002, in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

One common characteristic of all intelligent life is continuous perceptual input. A decade ago, simply recording and storing a a few minutes of full frame-rate NTSC video required special hardware. Today, an inexpensive personal computer can process video in real-time tracking and recording information about multiple objects for extended periods of time, which fundamentally enables this research.

This thesis is about Perceptual Data Mining (PDM), the primary goal of which is to create a real-time, autonomous perception system that can be introduced into a wide variety of environments and, through experience, learn to model the activity in that environment. The PDM framework infers as much as possible about the presence, type, identity, location, appearance, and activity of each active object in an environment from multiple video sources, without explicit supervision.

PDM is a bottom-up, data-driven approach that is built on a novel, robust attention mechanism that reliably detects moving objects in a wide variety of environments. A correspondence system tracks objects through time and across multiple sensors producing sets of observations of objects that correspond to the same object in extended environments. Using a co-occurrence modeling technique that exploits the variation exhibited by objects as they move through the environment, the types of objects, the activities that objects perform, and the appearance of specific classes of objects are modeled. Different applications of this technique are demonstrated along with a discussion of the corresponding issues. Given the resulting rich description of the active objects in the environment, it is possible to model temporal patterns. An effective method for modeling periodic cycles of activity is demonstrated in multiple environments.

This framework can learn to concisely describe regularities of the activity in an environment as well as determine atypical observations. Though this is accomplished without any supervision, the introduction of a minimal amount of user interaction could be used to produce complex, task-specific perception systems.

Thesis Supervisor: W.E.L. Grimson
Professor of Computer Science and Engineering

Acknowledgments

I would like to express my gratitude to the Artificial Intelligence Laboratory for the environment of freedom and creativity that allowed me to pursue my (sometimes fanciful) dreams and ideas. It is difficult to imagine a better environment for the pursuit of artificial intelligence or growth as a researcher. These past seven years have helped to shape who I am, and I will always look back on them with the utmost fondness.

In particular, I would like to thank my advisor Eric Grimson in that regard. His guidance and unwavering support gave me the confidence to pursue my particular approach to computer vision on a large scale. He maintained the perfect balance between helping me develop my research and helping me develop myself as a researcher while always putting me and his other students interests first. I would consider myself fortunate if just a small portion of the kind of researcher and the kind of person he is has rubbed off on me.

My thesis committee was tireless. I am indebted to them for the effort they put forth in even the busiest of times. Apart from his useful feedback on this document, Thomás Lozano-Pérez advised me in my early days in the HCI room where I first began tracking objects in real-time. Rod Brooks interest in situated learning systems helped me define my research agenda and his comments were also a factor in the development of this thesis.

One of the most influential forces in my career has been my advisor at Northwestern University, Paul Cooper, who gave me a start in Artificial Intelligence and Computer Vision. I would be on a different path if our roads had not crossed. I would also like to thank Mr. Downey, Dan and Katie Boggs, Gary Harpster, my Aunt Ellen, my Grandmother and her sisters, Mr. Aura, Ryan Clausman, and so many others. These people embody many of the characteristics that I strive to achieve every day of my life.

At the AI Lab, those I'd like to thank are too numerous to even list. Kinh Tieu, John Fisher, and particularly Erik Miller have played major roles in helping me define my approach to computer vision. I place great value in the helpful discussions with Mike Leventon, Mike Ross, Greg Galperin, Jeremy DeBonet, Jim Glass, and Christian Shelton. Those who have contributed more directly to the concepts in this thesis are: Erik Miller, Kinh Tieu, Gideon Stein, Raquel Romano, and Lily Lee.

Beyond supporting me in uncountable ways, this thesis would not exist but for the gifts my family gave me. My father instilled in me a thirst for knowledge and a desire to make a difference. I credit my mother with my perseverance and tenacity. My sister, my brother, his wife, and my two nephews have taught me about balance in life. And finally, for her love, her trust, her patience, and for reminding me to eat, I am forever grateful to my best friend and my love, Parul.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the main concepts underlying Perceptual Data Mining and details the structure of this document. The first section describes the goals of this research. Section 1.2 discusses some guiding principles that led us to attack this particular problem and to use Perceptual Data Mining in this capacity. Section 1.3 introduces our world model, which lays the groundwork for a description of each component of our system in Section 1.4. Because the breadth of this document may be more than most readers require, the final section describes different paths through this document that may be appropriate for readers with different interests.

1.1 Goal

The primary goal of Perceptual Data Mining (PDM) is to create an autonomous perception system that can be introduced into virtually any environment and, through experience, learn to model the active objects of that environment. We would like to infer as much as possible about the presence, the type, the identity, the location, the appearance, and the activity of each active object in an environment from multiple video sources, without supervision. We choose to initially neglect the static elements of the environment (those that never move) because they are often the least interesting and they are generally more difficult to model because they do not exhibit any variability.

Perceptual Data Mining (PDM) is a bottom-up, data-driven approach that models regularities in the representations of active objects. By taking advantage of general assumptions about the state of the active objects, complex models of the sensors, the environment, the objects, and the activities of the objects can be built. For example, assuming that the state of an object varies smoothly enables tracking from frame to frame. Assuming that an object can only occupy one position in space at any particular time can allow for various levels of camera calibration and correspondence matching. Assuming that the class of a tracked object does not change enables improved clustering based on class.

To begin the process of modeling the active objects, a basic attention mechanism that reliably detects the objects of interest is required. Beginning with only this basic

attention mechanism, it is possible to bootstrap one capability from another until the system reaches a limit that it cannot surpass without explicit user guidance (supervision). Reaching this point is the ultimate goal of PDM. Once this state is reached a minimal amount of user guidance (supervision) or user interaction (communication) can produce complex, task-specific perception systems.

Based on the assumptions we have made, we can detect moving objects. We can establish correspondence between those moving objects through time and across multiple sensors. We can normalize properties of the tracked objects in certain environments. We can cluster the type of object based on object shape. We can cluster the activities that objects perform. We can learn cycles of repeated activity in a site. We can model large environments. We can find anomalies based on any aspect that we have modeled. We are capable of doing all this in a wide range of different environments without any explicit supervision.

1.2 Guiding principles

Because this thesis describes a very complex system with many parts, it can be difficult to understand exactly why a particular choice was made in designing the system. While these specific choices will be described throughout this thesis, we list below three guiding principles that have been employed throughout this research:

Data-driven modeling

When we first began tracking objects 24 hours a day/seven days a week, we were amazed by the amount of data we were able to collect. In a few minutes, we collected more data than the COIL-120 database [39] commonly used for object recognition experiments. In a few hours, we collected more data than the FERET database [45] used for face recognition. In a single camera in one morning, we collected more data than the COREL database [5] used for image indexing experiments. By tracking in multiple cameras for more than four years, we have collected hundreds of terabytes of data[1]. This quantity of data is daunting but also enabling.

Perceptual data is grounded in real objects

Further, the data we collect is more structured than a random collection of images. Our data is observations of real objects in the world over time. Notably, this type of data is available to all situated vision systems, whether biological or computational in nature, throughout their development. Though the identity of tracked objects is not known, we usually make tens to hundreds of observations of the same object. We also see similar activities performed by objects of the same class. There is also temporal structure to the types of objects and activities. This thesis examines the many ways of leveraging this type of data.

[1]Unfortunately, only a small portion of this data remains due to storage constraints.

Maximize applicability by minimizing restrictive assumptions

Building a vision system by assuming a particular task and embedding as much knowledge as possible about that task into the system allows one to build very capable systems. For instance, manually trained face recognition systems currently perform better than a human at certain tasks. This is a valid approach, but our primary goal is to create broadly applicable systems that extract their models from the data they observe. Every assumption that is made reduces the applicability of the system to situations where that assumption is (mostly) valid. The primary requirement in producing a general vision system is limiting the assumptions that are made about the active objects and their environment.

Of course, some assumptions are required to learn anything, but our goal is to use the most general assumptions possible. We assume that we are seeing the world through multiple static cameras. We assume that active objects' visual representations tend not to overlap and their visual representations tend to differ from the static objects that appear in the same location. We assume that objects' dynamics tend to vary smoothly in time. We assume that tracked objects maintain their identity. Other assumptions will be detailed throughout this thesis. These assumptions limit our system to situations where these assumptions are valid, but we will show by example throughout this thesis that these assumptions are very general.

What differentiates this approach from similar approaches is what we do not assume. We do not assume that any supervision is available. We do not assume the number of objects is known. We do not assume the objects are of a particular type or small set of types (e.g., people or vehicles). We do not assume the types of objects are known. We do not assume the number of types of objects is known. We do not assume the geometry of the world is known. We do not assume there is visual overlap between cameras. In cases where there is overlap, we do not assume the geometric relationship between the cameras is known.

1.3 Our model

In this section we introduce a model that can be used to describe many problems in computer vision including tracking, correspondence, clustering, and classification. While some important aspects of computer vision are not covered by this model, it enables a concise description of a large portion of the field. In particular, it is useful in articulating all major components of this thesis. It is also useful in understanding where this work is situated within the field of computer vision.

Our model of the world is composed of a set of N objects, S_X, and a set of M (imaging) sensors, S_I

$$W = (S_X, S_I). \tag{1.1}$$

In order to model all aspects of computer vision other state would be required (e.g., lighting sources), but this model is expressive enough to discuss a majority of computer vision research.

The set of objects can be modeled by a set of object state sequences through time

corresponding to the N objects, $\{X_1, X_2, ..., X_N\}$ where object state is described below. The set of sensors can be modeled by a set of sensor state sequences through time corresponding to the M sensors, $\{I_1, I_2, ..., I_M\}$ where sensor state is described below. Each object has a state at each point in time

$$X_i = (X_i(1), X_i(2), X_i(3), ..., X_i(T)) \tag{1.2}$$

and each sensor has a state at each point in time

$$I_i = (I_i(1), I_i(2), I_i(3), ..., I_i(T)). \tag{1.3}$$

We further factor the state of object i at time t into

$$X_i(t) = (U_i, L_i(t), p_i(t), c_i(t), s_i(t)) \tag{1.4}$$

where U_i is a unique label for a particular object (which does not vary over time), L_i is a class label or set of class labels, $p_i(t)$ is the three dimensional position of the object, $c_i(t)$ is the configuration of an object (description and relative positions of the parts), and $s_i(t)$ is the surface properties of an object. Class labels can include object type (e.g., person, car, trash, leaves, etc.), object activity (e.g., walking, running, roller blading, etc.) or any other type of classification. This is not the meant to be taken as the only factorized representation of object state, but it is useful in the context of this thesis.

Currently, we are considering only cameras as sensors. Hence, we factor the state of sensor j at time t into

$$I_j(t) = (t_j(t), r_j(t), i_j(t)) \tag{1.5}$$

where $t_i(t)$ is the position (translation) of the sensor, $r_i(t)$ is the orientation (rotation) of the sensor, and $i_i(t)$ describes the intrinsic parameters of the sensor. Intrinsics for cameras can include imaging sensor size, aspect ratio, focal length, auto gain control, aperture, and many other properties of the sensor.

At a single point in time, the state of the world can be fully described by the state of the objects and the sensors at that time

$$W(t) = (X_1(t), X_2(t), ..., X_N(t); I_1(t), I_2(t), ..., I_M(t)). \tag{1.6}$$

Given the state of an object and a sensor at time t, it is possible to determine the instantaneous, sensor-relative (observable) state of the object i in sensor j with a deterministic function, f.

$$O_i^j(t) = f(I_j(t), X_i(t)). \tag{1.7}$$

The imaging process for sensor j is simply a function producing an image at time t, $Y^j(t)$, given the sensor-relative observable state of the objects in the world at that time

$$Y^j(t) = g(O_1^j(t), O_2^j(t), ..., O_N^j(t)). \tag{1.8}$$

where g is a complex function which produces an image while taking into consideration all the visibility constraints of all the objects. In this model, background scene elements such as sky, ground, or walls are also considered objects.

Computer vision and computer graphics are often posed as inverse problems. Computer graphics can be described as producing the images (Y^j's) given the state of the world. The field of computer vision is the inverse–trying to infer the state of the world from the images.

1.3.1 The landscape of computer vision

Many problems in computer vision can be effectively described as "estimating the state of the world $(W(1), W(2), ..., W(T))$ from a set of images $(Y^j(1), ...)$". The set of images may be a sequence, a set of images from different sensors at the same time, or images with no regular relationship. Apart from some low-level vision problems, the model introduced above can be used to describe many computer vision problems though the granularity of the definition of "object" can vary considerably (e.g., moving objects, edges, corners, skin-colored regions, etc.). Different bodies of research make different assumptions about the world and try to infer different aspects of the state of the world. Below we will describe a few major areas of research in computer vision to aid in the understanding of the model introduced above (without attempting to exhaustively describe the field).

- *Egomotion*- Egomotion attempts to model the state of the sensor or sensors, $\{I_j(t), ...\}$, and often assumes the objects are static $(\{X_i(t_1) = X_i(t_2) \forall t_1, t_2\})$. Often the "objects" are assumed to correspond to simple features in the images (e.g., corners, lines).

- *Tracking*- In contrast to egomotion, tracking tries to estimate the state of the objects, $\{X_i(t), ...\}$, and often assumes the sensors are static $(\{I_i(t_1) = I_i(t_2) \forall t_1, t_2\})$. If not, an attempt is often made to estimate and factor out that state. Two major types of tracking are:

  - *Far-field tracking*- Far-field tracking is primarily interested in extracting the position of many objects in situations where the objects are far from the sensor and usually do not visually interact. Here the most significant problems are determining the number of objects and learning correspondence between multiple observations of the same objects.
  - *Near-field tracking (articulated tracking)*- Near-field tracking is primarily interested in modeling the configuration of the object over time. It is often assumed that there is at most one object and the object is always completely visible.

- *Object classification (Supervised classification)*- Supervised classification attempts to assign class labels to unlabelled observations, $O_x(t)$. These systems are trained on large sets in which the unique labels, U_i, are provided.

- *Object clustering (Unsupervised classification)*- Unsupervised classification often takes sets of observations, $O_i(t)$, without information about the object's class labels L_i, and attempts to determine a set of object labels that correlate with the true classes labels. It is difficult to evaluate the descriptive power of the labels determined by such systems except in the context of a supervised task.

- *Reconstruction*- Reconstruction work usually assumes observations from more than one sensor, $\{Y^1(t), Y^2(t), ..., Y^M(t)\}$. Some work attempts to model both the relative state of the sensors and the state of the objects in the world.

  - *Stereo*- Stereo assumes knowledge about the relative position of the sensors. It attempts to model the correspondence of every region in one scene to regions in the second scene. Recently, this has been used as a pre-processing algorithm for tracking.

  - *Image mosaicing*- Mosaicing works on the assumption that multiple cameras are roughly at the same position and a projective homography of the images can be used to bring the features (edges, corners, pixels, or image regions) of the image into correspondence.

- **Perceptual Data Mining**- The goal of Perceptual Data Mining is to infer as much as possible of the state of the objects and the sensors in the world from continuous visual observation in static cameras.

  - Given a sequence of images ($Y^j(1)$, $Y^j(2)$, $Y^j(4)$, ..., $Y^j(t)$), our visual attention mechanism determines a set of active object observations in each frame of each sensor. Each observation is given a unique label. E.g., $\{o_1, o_2, o_3, ...\}$.

  - By modeling the sensors, environment, and object dynamics, our correspondence system can refine the estimates of object position and velocity to a more global frame of reference and can determine sets of object observations that are likely to have the same unique label (U_i). A set of observations that correspond to the same object in different sensors or at different times is referred to as a *Multiple Observation Sets* (MOS). E.g., $M^u = \{O_u^{j_1}(t_1), O_u^{j_2}(t_2), ...\}$

  - Multiple Observation Learning (MOL) clusters observations while exploiting the variability exhibited by the active objects in the environment. MOL was used to determine latent class models for object shape and object activity by clustering based on those aspects of the object observation description. These latent class labels are likely to be predictive of the true class labels, $\{L_i\}$. MOL can also be used to determine a factored representation of images of a class of objects and to determine tissue type variation in unsegmented magnetic resonance images.

  - This complex descriptive model of the types of objects and the activities that they perform can be exploited to classify MOSs, determine anomalous

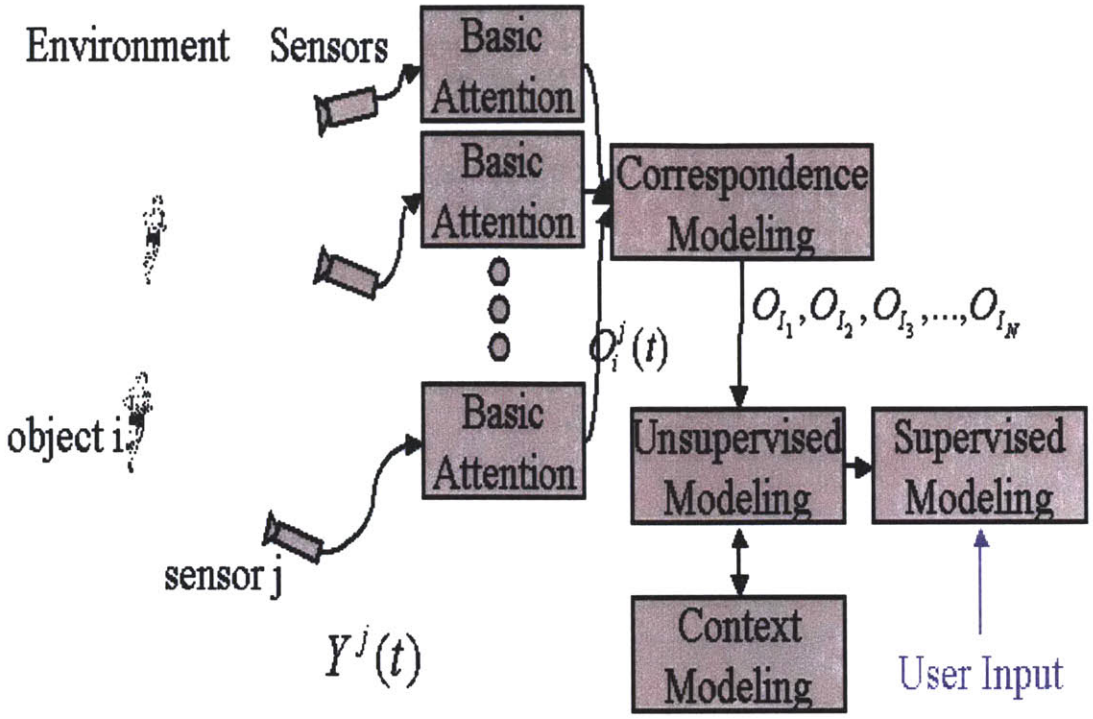


Figure 1-1: *This figure shows the system architecture for our perception system.*

MOSs, and model temporal structure of the activity in the environment. This rich, complex description of an environment shows promise for numerous applications and customization to particular tasks.

1.4 Outline of Perceptual Data Mining

Shortly after birth, infants develop the ability to track moving objects by combining the ability to saccade to areas of interest and a sub-cortical mechanism for stabilizing visual input. We have developed a system with a similar function that tracks moving objects in an environment. In this thesis, we show that it is possible to bootstrap an entire visual processing system from this basic attention mechanism.

Figure 1-1 shows the architecture of the complete learning system. The basic attention mechanism on each sensor in the environment detects the presence of active objects at each point in time. Correspondence modeling groups these separate observations based on their identity. The groups are used to build unsupervised representations of the objects in the environment. At the end of the process, user guidance and user interaction can be used to associate the learned representation with task-specific labels.

This section outlines each component of the system in the context of a particular example, tracking the activity around the Artificial Intelligence Laboratory in Cambridge, Massachusetts. This example was chosen because it is our longest experiment

and has the most comprehensive results. In the following chapters other examples will be given to illustrate the generality of our approach. After reading this section, one should have a basic understanding of the entire system.

1.4.1 The environment and the sensors

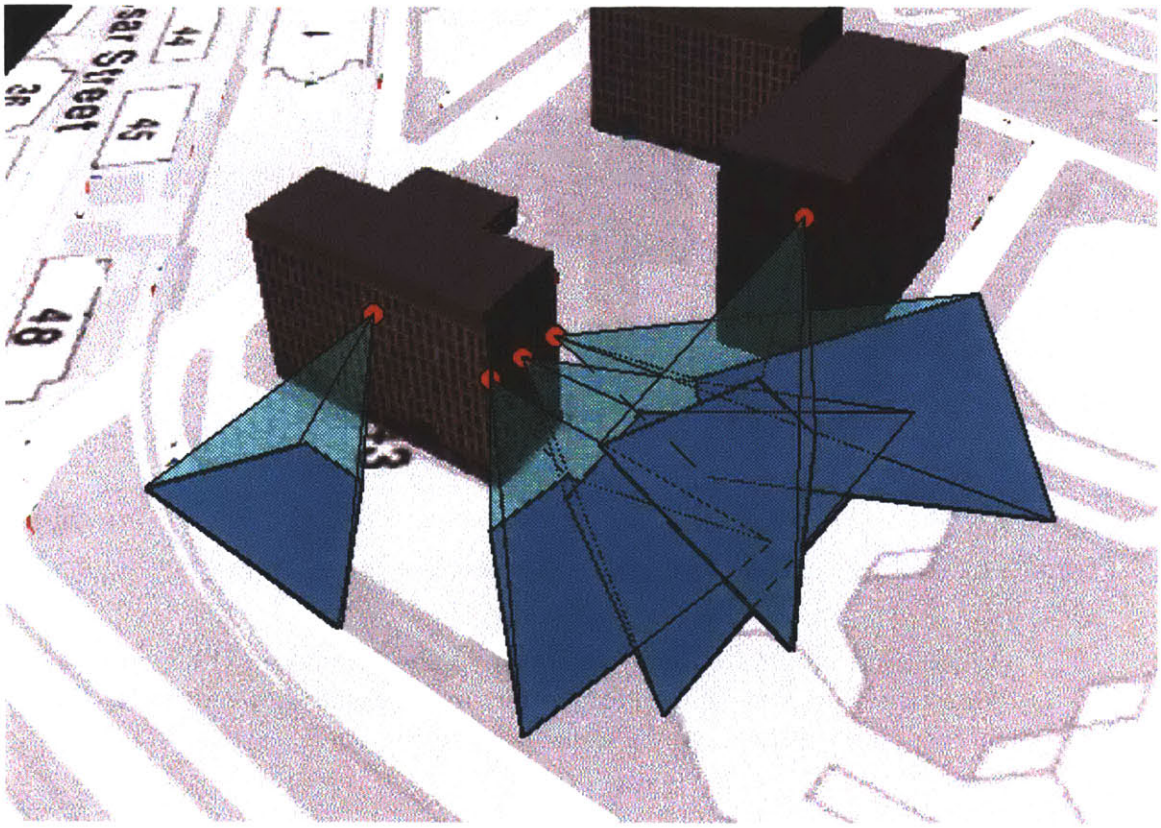


Figure 1-2: *Example camera locations for our experimentation around the Artificial Intelligence Laboratory in Cambridge, Massachusetts. Five camera positions and the region of space they can view are displayed. We would like to thank the MIT Computer Graphics Group for this reconstruction of Technology Square.*

We have chosen to explain the components of the system in the context of our longest running experiment, modeling the activity around 200 Technology Square[2]. In a single day, thousands of people and vehicles as well as various other types of objects move through this environment.

For more than four years, our system has been continuously monitoring the moving objects around the MIT Artificial Intelligence Laboratory using multiple cameras. Figure 1-2 shows an example positioning of a set of cameras and the region of space they can observe. This example of camera placement illustrates that some cameras overlap while others do not. Our system is robust to the number of cameras and the

[2]This building was 545 Technology Square when this experiment began.

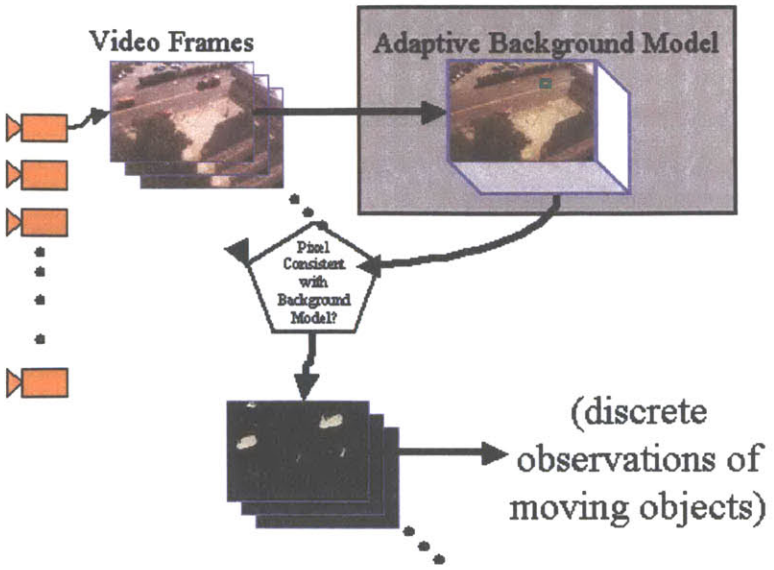


Figure 1-3: *This figure shows the process of background estimation. A series of video images are used to estimate the appearance of the static background. This model is used to determine the pixels that are not consistent with that model. Those pixels are grouped to determine a discrete set of moving objects.*

camera placement. While our system requires that the cameras are static, our system does not require any information about the placement of the cameras or their visibility constraints. Each camera j produces images at discrete points in time, $Y^j(t)$. This set of video streams is the only input into our system.

1.4.2 Basic attention (Chapter 2)

A robust, general attention mechanism is required for all subsequent research in Perceptual Data Mining. Our visual attention mechanism attempts to detect all visible, moving objects in static scenes by modeling the appearance of the non-moving elements as seen through a static camera. In the AI Laboratory scenario, examples of moving objects are people, vehicles, trash, trees blowing in the wind, and various animals. Examples of non-moving objects are patches of grass, cement, buildings, and parked cars. To determine which pixels are likely to have resulted from the presence of a moving object, our method uses a novel variant of background estimation.

The goal of background estimation is to detect a small set of likely observation states of visible, active objects, $\{O^j(t), ...\}$ for each frame taken from each sensor, $Y^j(t)$. No information about the identity of these observed objects is available. The only available information is the time, sensor, and the characteristics that can be determined by the attention mechanism. In our case, the characteristics we derive are the image-relative centroid, size, width, height, projected silhouette, and projected appearance.

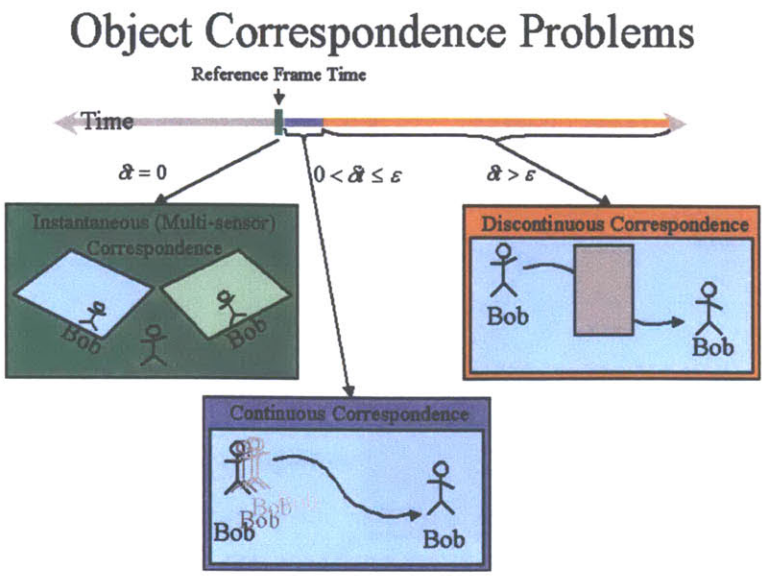


Figure 1-4: *This figure shows the three essential problems in establishing object correspondence. Instantenous object correspondence is correspondence between pairs of observations of the same object across sensors. Continuous object correspondence is correspondence over time. Discontinuous object correspondence is correspondence after missing observations caused by distraction or objects leaving the environment.*

The bulk of Chapter 2 centers on the complexity of modeling the static background using our background estimation technique. Figure 1-3 shows the general framework of background estimation.

Our technique uses online estimation of an approximation to a mixture of Gaussians for each pixel. The two major advantages of this implementation are quick recovery when stopped objects move and some robustness to repetitive background changes. It allows for effective tracking in reasonably sparsely populated environments. The processing required to track the detected foreground objects in multiple cameras is discussed in the following chapter.

1.4.3 Modeling object correspondence (Chapter 3)

A multiple observation set (MOS) is a set of observations of the same object in the world, either taken at different times or from different sensors. An ideal object correspondence system would establish N multiple observation sets (MOSs) that correspond to observations of each object. Each set would contain all observations of a single object regardless of when it occurred or which sensor made the observation. Every car, every person, every piece of trash would have a single corresponding MOS. Nothing about the true identity of the MOS would be available, but every observation in each MOS would correspond to a single object. Whenever that object appeared in any sensor, the system would assign it to the proper MOS. This can be posed as a labeling problem.

Without perfect tracking in a completely observable system or user supervision in a restricted domain (e.g., face recognition with a limited domain), there is little hope of achieving this ideal goal. Fortunately, there are regularities that can be exploited. Figure 1-4 shows the three essential problems in establishing object correspondence on a timeline. Chapter 3 discusses how to exploit these regularities.

First, if an object is detected in a location at the same *instant* by multiple sensors, instantaneous object correspondence can be established. There are three possible correspondence relationships between a pair sensors including: having no region of visual overlap in which objects are detected; having an approximately planar set of corresponding points in the area of visual overlap; and having correspondences that span a full three dimensional subspace in the area of visual overlap. We cover models for each of these cases as well a mechanism for determining which model is appropriate in a given situation.

The most common and (most difficult to detect) case is when there is no visual overlap. When there is visual overlap, the data often lies on one or a set of planes. The approximately planar case can be fit by single homographies or multiple homographies. The three dimensional case may enable Euclidean reconstruction of the tracking data. Even in this case, determining a set of correspondence planes can be useful in establishing object correspondence. Though much work has been done on multiple sensor correspondence, tracking correspondence has some peculiarities that can be exploited. We will fully discuss these peculiarities and how they can affect this type of reconstruction.

Second, *continuous* tracking in individual sensors (or unified sensors) should enable many detections of the same object to be put into an equivalency class. Approaches vary from maximum likelihood tracking to full density estimation through time. We determined that a multiple hypothesis tracking system suited our computational and robustness requirements best.

Third, discontinuous correspondences are investigated. Short-term discontinuities can result from interactions between tracked objects or various types of distraction. We discuss methods of exploiting models of dynamics, appearance, and site-specific behavior models to re-establish object correspondence in these cases. Long-term discontinuities can result from occlusions, lack of site coverage, or objects leaving the environment for a period before returning.

Figure 1-5 shows an application of these techniques in the context of the 200 Technology Square monitoring experiment. Given the tracking data from three different cameras, a correspondence model can be estimated. Using the correspondence model allows one to track objects through multiple sensors in extended environments.

Using these systems in multiple environments over the period of more than four years, we processed terabytes of data including billions of images and movements of tracked objects. Each set of observations of the same object is termed a Multiple Observation Set (MOS). The following chapter discusses methods for exploiting MOSs.

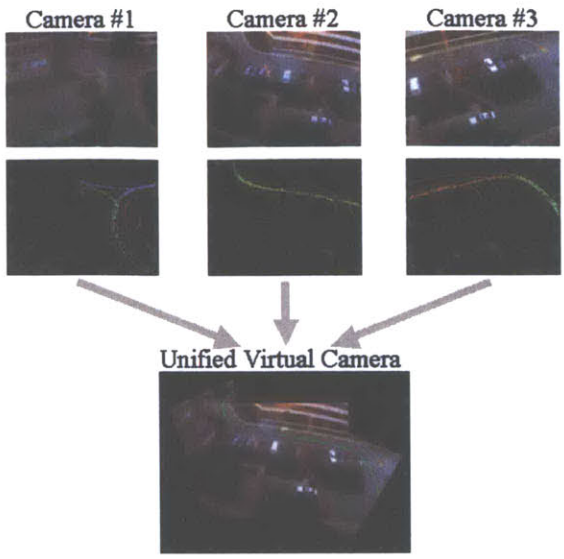


Figure 1-5: This figure illustrates how a correspondence model of three overlapping cameras allows an object to be tracked through multiple sensors in an extended environment.

1.4.4 Co-occurrence based clustering (Chapter 4)

Having established hundreds of thousands of Multiple Observation Sets (MOSs), it is possible to determine models of observations for classes of objects by exploiting variability in observations we have recorded of the same underlying objects. Chapter 4 discusses Multiple Observation Learning (MOL), a broadly applicable method of co-occurrence-based non-parametric modeling of phenomena which is directly applicable to multiple observation sets (MOSs) and other co-occurrence problems.

A landscape of classification and clustering problems is introduced to better understand the relationship between MOL and other clustering and classification systems based on the amount and type of supervision required for each. Figure 1-6 shows that MOL requires no explicit labeling supervision and a modest amount of relational (pairwise identity) supervision that are acquired at little cost. With the addition of minimal labeling supervision, an MOL system could achieve an acceptable level of performance at a particular task while minimizing the associated human cost.

Using a simple example involving pairs of numbers from one to ten produced independently and identically distributed (IID) from a number of people, a generative latent class model is introduced. This generative model involves choosing a latent class (person) with some probability and then choosing multiple observations (numbers from one to ten) from that latent class' probability mass function (pmf). In our simple example, the chance of choosing an individual is their likelihood of participating in the test and their pmf describes their likelihood to choose the numbers from one to ten. The resulting co-occurrence matrix describes the probability of a pair of observations

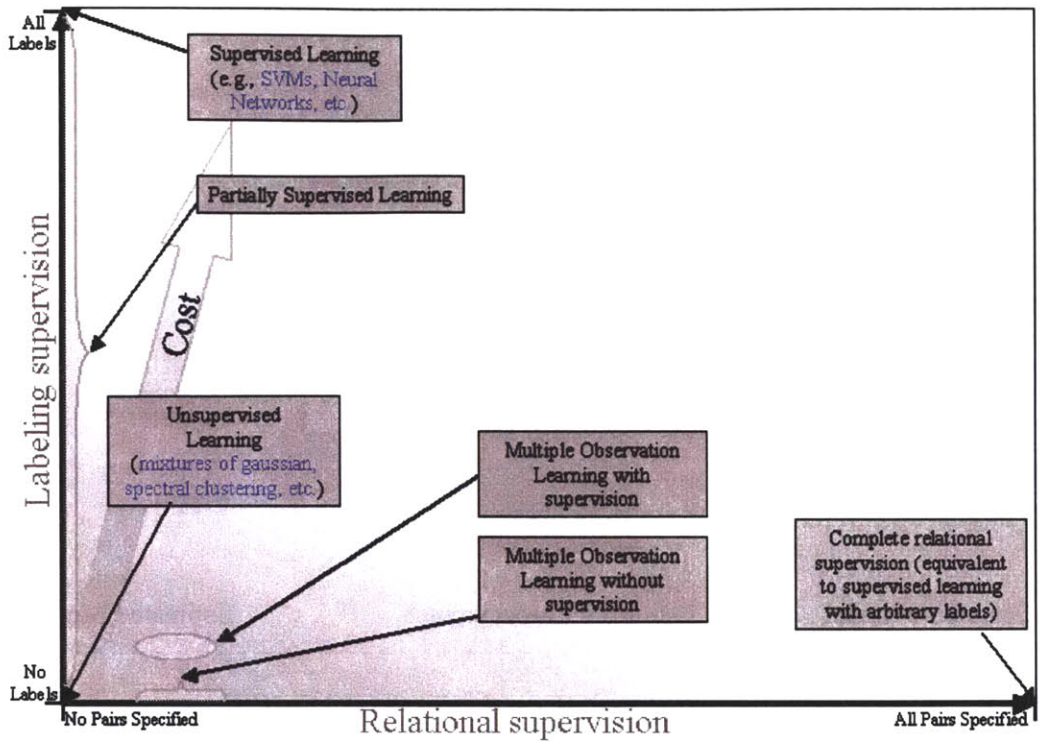


Figure 1-6: This figure shows relational supervision vs. labeling supervision.

occurring in any MOS.

$$\hat{C}_{i,j} = \sum_{c=1}^{N} \hat{p}(c)\hat{p}(i|c)\hat{p}(j|c). \tag{1.9}$$

Figure 1-7 shows two simple examples involving a single person and a set of three individuals with different preferences on choosing values from one to ten. Given a set of pairs of observations from individuals the corresponding co-occurrence matrices is estimated. The goal of MOL is to determine parameters of a latent class model which is consistent with the estimated co-occurrence given the proper number of latent classes. With relatively clean data, the number of latent classes can also be determined. This chapter outlines the process of accumulating co-occurrences, estimating the latent classes, and classifying MOSs. Then some of the assumptions and considerations of MOL in discrete spaces are discussed.

One of the most significant considerations in MOL is computational and storage requirements. These are a fundamental limitation for MOL in large or continuous observation spaces. Thus, methods of applying MOL in continuous observation spaces are discussed including three types of discrete tilings of the spaces: uniform tiling; density-based tiling, and tiling based on Associative Mixture of Gaussians estimation. Different factors that affect performance are investigated through five sets of experiments. The generality of this mechanism is illustrated in four example applications.

Using different aspects of the description of the objects it is possible to determine

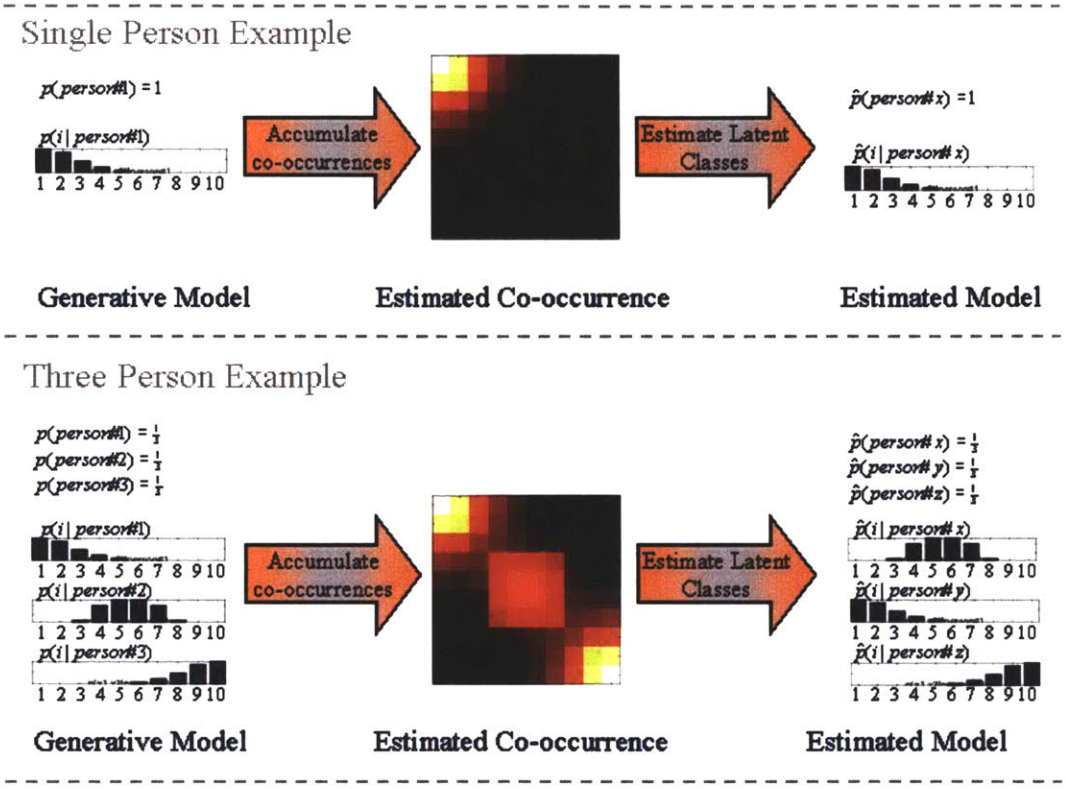


Figure 1-7: This figure shows two simple examples. Each shows some latent class models on the left, a co-occurrence matrix (brighter values are larger in the center, and the estimated class models on the right. For the three person example, there is only one set of latent class model parameters that are consistent with the observed co-occurrences.

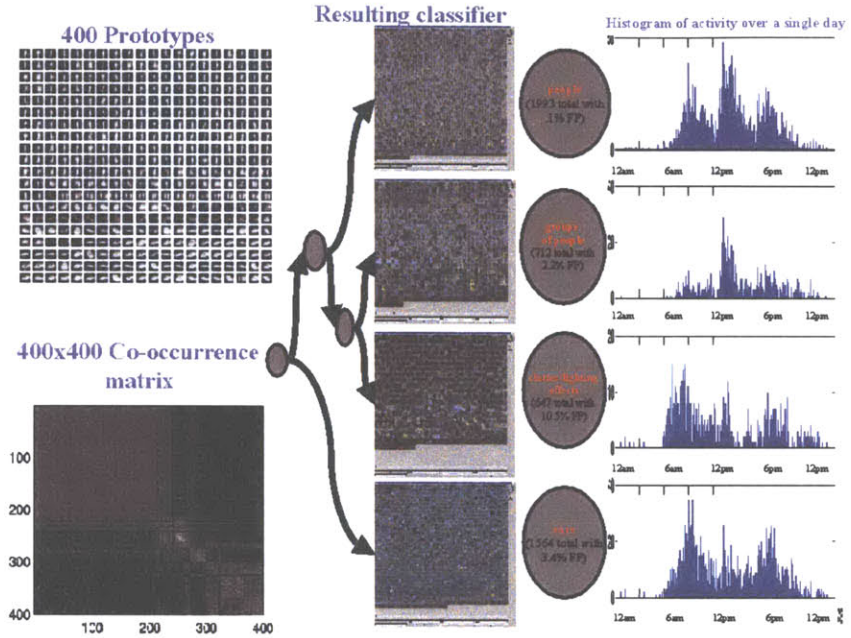


Figure 1-8: *On the left is the 400 silhouette prototypes and the co-occurrence matrix that resulted from a day's worth of tracking sequences. In the middle is the classification hierarchy which resulted, images of all occurrences of each class, and description of the classes as well as their performance relative to those descriptions. On the right are 24 hour histograms of the occurrences of each class. For higher quality images, see: http://www.ai.mit.edu/projects/vsam/.*

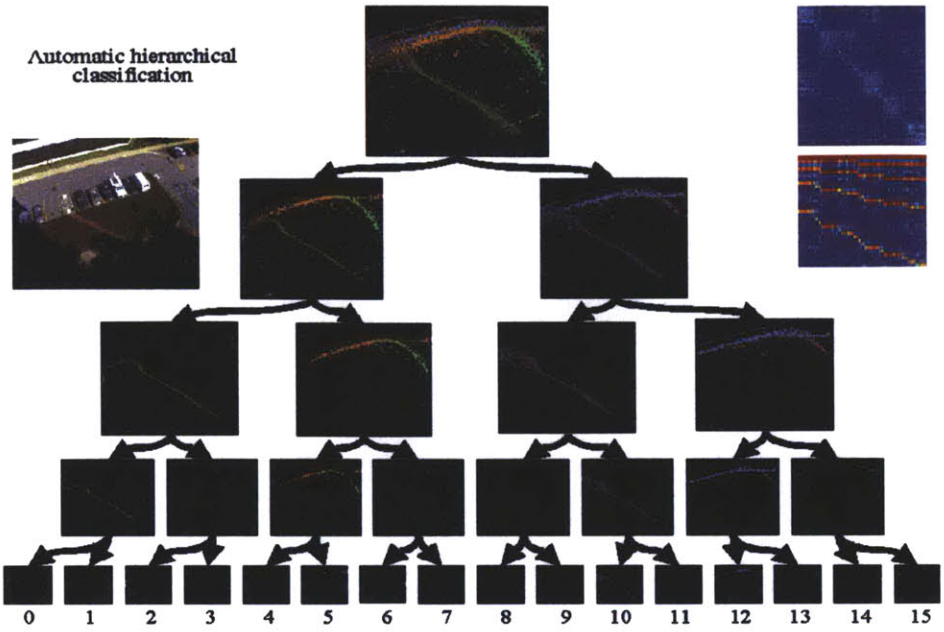
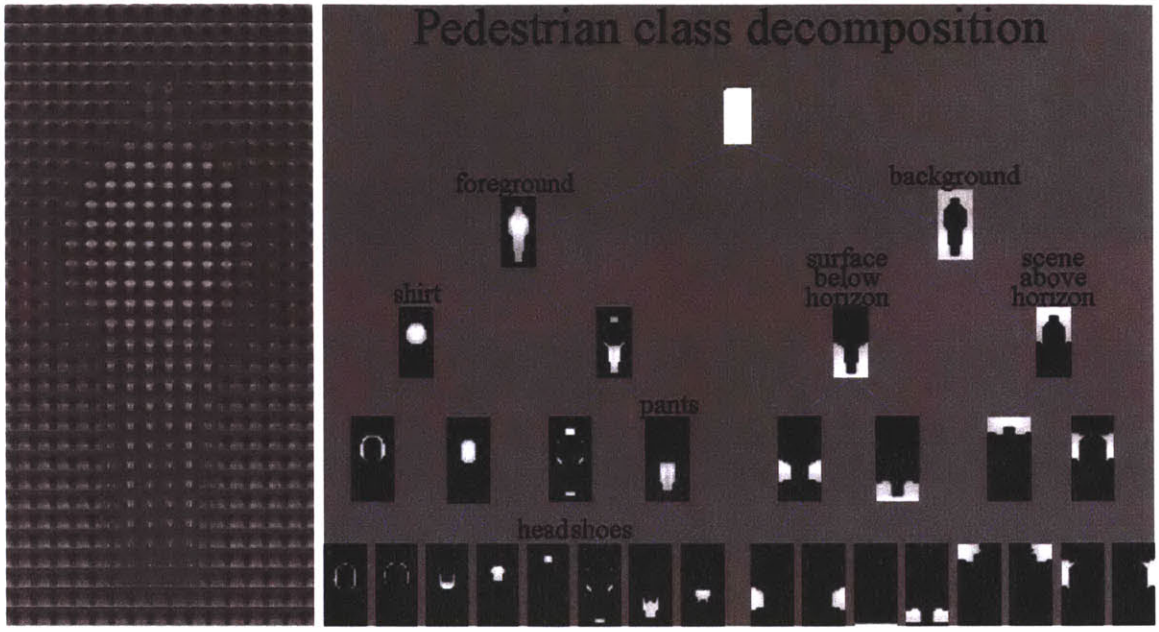


Figure 1-9: *This figure shows an image of the scene(upper left), the classification hierarchy(center), and the co-occurrence matrix and normalized pmfs(upper right) for each element of the tree. The scene contains a road with adjacent parking spots and a path through the grass near the loading bay of our building. The binary tree shows accumulated motion templates for each node of the tree. And the co-occurrence matrix and normalized pmfs show which prototypes occurred within the same sequences and the probability distributions for each node in the tree (ordered breadth-first). The final level of the tree specific classes including: pedestrians on the path (one class in each direction); pedestrians and lawn-mowers on the lawn; activity near the loading dock. cars; trucks; etc. These classes can be viewed in a Java 1.1 compatible browser at:* `http://www.ai.mit.edu/projects/vsam/Classification/Cclasses/` . *Note: the columns and rows of the co-occurrence matrix have been ordered to make some of its structure more apparent.*

different classifiers based on appearance and activity. The first application determines a hierarchical clustering based on silhouette shape, which describes the types of objects in that environment. Figure 1-8 shows an example of classification based on shape. It effectively clusters the observations based on shape into vehicles, individual pedestrians, groups of pedestrians, and clutter and lighting effects. This is a concise description considering the variability in object shape due to different positions, angles, sizes, and configurations of these classes of objects.

The second application determines a hierarchical description of the types of activities in an environment. Figure 1-9 shows an example of hierarchical activity classification. Objects are first differentiated based on their direction of travel because few object changed directions. Then, the path and road traffic are separated because there was not significant overlap in those two activities. A tree of depth four results in a concise description of the different significant clusters of activity in this



(a) (b)

Figure 1-10: The similarity template and the corresponding automatically generated binary decomposition of the images in the pedestrian data set. The root node represents every pixel in the image. The first branch splits foreground vs. background pixels. Other nodes correspond to shirt, legs, head, and background regions.

environment.

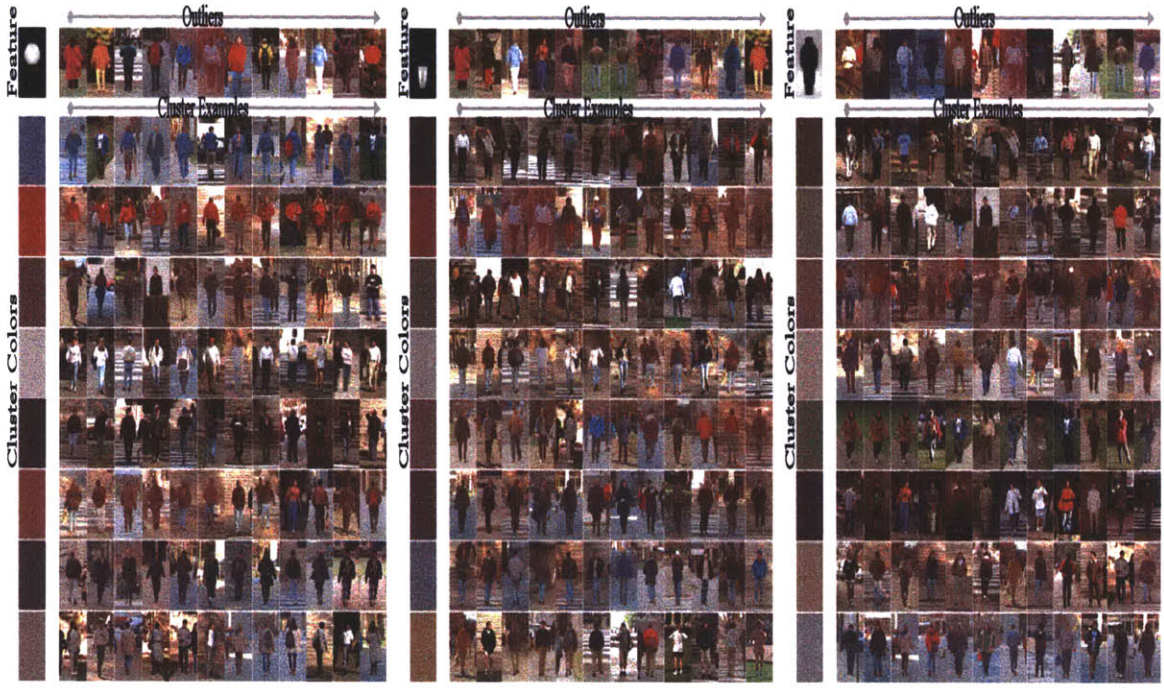
Next, the Similarity Template (ST), a new image representation that models likelihood of pixels resulting from the same region, is introduced. We use STs to estimate a *static* visual attention mechanism from our attention mechanism based on motion. This static attention mechanism could eventually enable PDM to locate and track objects similar to the active objects in its environment in non-static cameras (e.g., on the web) based on its experience in an active environment.

The third application of MOL determines a hierarchy of component regions for a set of images of a single class of object based on the Similarity Template. Figure 1-10 shows a similarity template for a pedestrian data set and the resulting decomposition into component regions. Figure 1-11 shows automatically derived clusters conditioned on different aspects of the model (shirt, pants, and background colors).

The final application determines latent tissue class models in magnetic resonance images given measures of local co-occurrence. The wide range of applications of MOL shows its promise from low-level modeling to high-level modeling.

1.4.5 Meta modeling (Chapter 5)

Thus far, the thesis has illustrated the ability to characterize models of the environment, the sensors, the type of objects, the activities of the objects, and even characteristics of the appearance of particular types of objects. Chapter 5 discusses methods



(c) (d) (e)

Figure 1-11: Results of automatic clustering on three components: shirt, pants, and the background. Each shows the feature, the most unusual examples of that region, followed by the 12 most likely examples for the eight prototypical colors of that region.

(a) (b)

Figure 1-12: This figure shows the two environments used as examples in the cyclic context modeling experiments. The first is an office environment in which a refrigerator was placed that contained inexpensive soft drinks. Most of the activity in this area was graduate students using the refrigerator or individuals passing through the environment. The second example is a traffic intersection in Boston.

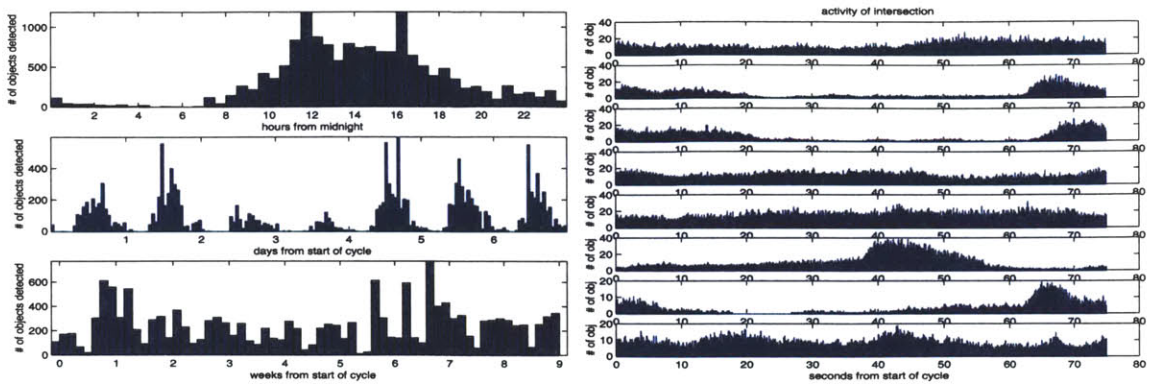
of further exploiting these automatically-derived, rich representations of activity in an environment.

Thus far, the time an object was observed has not been considered. In many environments, the temporal context of an event can be extremely important. We introduce a system which first determines temporal context cycles and then exploits the context cycle to characterize the activity of the scene. Figure 1-12(a) and (b) show two example environments that contain periodic context cycles.

In the office setting, the system automatically determined that there are 24-hour and 7-day cycles of activities. Using this information we can characterize activities at unusual times of the day. We could also characterize days with unusual amounts of activity (e.g., vacation days and student visit weekends). In the traffic intersection, after determining eight characteristic activities in the environment, the system learns the traffic light cycle. This enables detection of events like people running traffic lights.

Figure 1-13(a) and (b) show the context-sensitive activity models for these two environments. In the office, daily and weekly patterns can be exploited to better model when activity is expected. The traffic intersection example not only illustrates another example of a context cycle, but the ability to articulate the periodic model for different classes. This would enable a system to model the types of objects, object activities, and object appearances expected at different times of day or days of weeks (or any other salient context).

One of the strengths of our probabilistic modeling technique is that not only can we characterize clusters of shape, activity, and appearance, but we can determine that certain MOSs are not characteristic of our model. Four types of anomaly detection are outlined: observation anomalies, co-occurrence anomalies, temporal anomalies, and anomalous activity periods. These anomalous observation sets are often of the most interest. E.g., traffic accidents, an elderly person falling down, a child climbing



(a) (b)

Figure 1-13: This figure shows wrapped histogram of activity for 24-hour and 7-day periods and a daily histogram for our office experiment. The amount of activity is shown in half-hour, one-hour, and one-day blocks respectively. The week period begins on at midnight on Thursday morning. The entire 62 day experiment lasted from February 13, 2002 to April 19, 2002.

a bookshelf or a tree, a delivery after midnight, traffic commuting patterns changed, etc.

1.4.6 Discussion and future work (Chapter 6)

Chapter 6 discusses the motivation and promise of the Perceptual Data Mining framework. Throughout the thesis, an attempt is made to advocate a general approach to modeling active elements of an environment. In some cases, specific functionality was sacrificed to achieve a more broadly applicable system. The beginning of this chapter is an attempt to remind the reader of the benefits of those sacrifices. This includes discussion loosely relating human capabilities and neurophysiology to Perceptual Data Mining.

While this thesis attempts to cover the Perceptual Data Mining framework in both depth and breadth, there is significant areas of future research related to or enabled by the PDM framework. Various improvements in attention and correspondence are outlined that would augment the capabilities and applicability of the entire PDM framework. Methods for developing more factored, transferable representation and incorporating other modalities that would enabled modeling of additional structure in an environment are briefly outlined. Methods for exploiting additional context that could further generalize the system to many non-periodic environments are also summarized.

Finally, various means of incorporating explicit supervision are enumerated. While we illustrate that an unsupervised system is capable of building a robust, descriptive model of the activity in an environment, such a system has limited application to specific tasks without supervision. Supervision can be used to evaluate an unsupervised systems representation for a particular task and even altering the representation to better suit a task. An operator can tailor an activity filter to preferentially return

important or atypical events. Supervision could be elicited by the system to communicate aspects of its representation.

An intriguing aspect of the PDM framework is its general applicability. Potential application areas include security, elder-care, child-care, wildlife monitoring, home monitoring, traffic statistics, and intelligent environments. Statistics can be continuously gathered without human intervention. These statistics can be used to determine a variety of types of anomalies relating to object type, object activity, object appearance, or statistics of activities in an environment.

1.4.7 Appendices

Appendix A has pseudo-code for an implementation of the tracking system. Appendix B has pseudo-code for estimating of planar homographies. Appendix C has details of the factorization of a joint distribution into a mixture of marginally independent distributions used in the Multiple Observation Learning.

1.5 Reader's guide

Because of the broad scope of this thesis, I have outlined paths for readers with three different primary interests:

Tracking and correspondence– Because of the burgeoning interest in this area of research, many researchers have an interest in creating a robust tracking system. If you are most interested in understanding our indoor/outdoor, multicamera, 24/7 tracking system, read Chapter 2, Chapter 3, Appendix A, and Appendix B.

Non-parametric unsupervised modeling– Readers that are most interested in the unsupervised learning methodologies of this work should read Chapter 4, Chapter 5, and Appendix C.

Data mining driven computer vision research– I have found that a great deal can be gained by understanding why people have chosen their approach. I believe it would be a mistake to fully understand the mechanics of a body of research without understanding why that methodology was used. If you wish to understand why I have chosen the path I have and why I believe it holds promise for general computer vision research, read this chapter, the introductions to each of the following chapters, and the final two chapters of this work. This should help you understand my perspective and how my research relates to the field of computer vision as a whole.

Keeping the different goals of different readers in mind, each component will be treated in turn. Rather than discussing theory, implementation, and results for the entire system, these will be handled for each of the major topics described above.

Chapter 2

Attention: adaptive background estimation

This work is founded on the relatively new capability of processing large streams of information continuously in real-time and storing a reduced representation of that data for later processing using off-the-shelf hardware. Less than a decade ago, special hardware was required to simply record and store a few minutes of full frame-rate NTSC video signal. Now, an inexpensive personal computer can process each image in real-time, allowing multiple objects to be tracked and the pertinent data to be stored for days or even months. In the near future, cheap devices should be capable of more complex visual processing and subsequent analysis of the data in real-time, making this an intriguing area for further research.

This and the following chapter show the mechanism we used to process video streams in real-time and extract descriptions about the moving objects in the environment. This chapter describes a novel type of adaptive background estimation. Adaptive background estimation attempts to model the static "background" of an image sequence. Usually the model of the background is of less interest than the pixels that are outliers to the model, called "foreground" pixels. These pixels tend to lie on moving objects. The "foreground" pixels can be grouped using a method called connected components [20] into connected regions. These regions usually correspond to single moving objects or groups of moving objects in the scene.

This chapter outlines how to determine a discrete set of potentially moving objects from each frame in a video sequence, but not how to track objects across time and across multiple cameras. Chapter 3 outlines how to establish sets of object observations that correspond to the same object in the world by tracking objects through time and across multiple sensors. Appendix A outlines an efficient, approximate implementation of the entire tracking system.

This chapter begins by discussing related work in the field of tracking with particular attention to related work in background estimation. Then our approach to online estimation of the background is outlined. This system takes images as input and produces a foreground/background mask as output. Our approach uses an approximation to a mixture of Gaussians to model the intensity values of each pixel. Then it determines which components of the mixture model are most likely to have

resulted from static background objects. The pixel values that do not match these background components are considered "foreground" pixels. Extensions of the video tracking system are discussed, including a brief description of bootstrapping a static attention mechanism from our motion-based tracking system, which is discussed further in Chapter 4.

2.1 Related Research

Our goal as stated in the previous chapter extends beyond producing a tracker that estimates some aspect of the state of a particular set of objects in the world. We intended to create a tracker that would enable a system to automatically model all the active elements of a particular extended environment. This goal dictated many of the decisions we made in producing our tracking system.

A visual perception system should be robust to whatever is in its visual field or whatever lighting effects occur. It should not depend on careful placement of cameras. It should be capable of tracking objects through cluttered areas, objects overlapping in the visual field, shadows, lighting changes, effects of moving elements of the scene (e.g. swaying trees), slow-moving objects, and objects being introduced or removed from the scene causing long-term changes. Thus, to monitor activities in real outdoor settings, we need robust motion detection and tracking that can account for such a wide range of effects.

Traditional tracking approaches have difficulty in these general situations. Our goal is to create a robust, adaptive tracking system that is flexible enough to handle variations in lighting, moving scene clutter, multiple moving objects and other arbitrary changes to the observed scene. The resulting tracker is primarily geared towards video surveillance applications that cover large areas in which a few objects visually interact.

2.1.1 The field of visual tracking

It is difficult to define tracking in the context described here. The most appropriate Merriam-Webster definition of track (verb) is "To observe or monitor the course of (aircraft, for example), as by radar." While this is sufficient to describe some types of tracking, we are currently interested in visual tracking. We state the problem of visual tracking as "establishing correspondence between multiple visual representations of the same object."

This definition implicitly highlights two fundamental problems in computer vision. First, you must determine the visual representations that correspond to objects. Second, you must determine correspondence between these visual representations. Hence, we have chosen to conceptually divide the process of visual tracking into two separate tasks, object detection and object correspondence. Object detection involves determining likely states of objects (or the likelihood of states) given the observations. Object correspondence involves determining which sets of observations correspond to the same objects in the world.

Below we will begin to describe the vast variability in this field of research. We have highlighted in bold font the characteristics of our system:

- *modeled aspect of observable state-* There is a large variability in the aspect of the observable objects that can be modeled.
  - *image-based models-* e.g. face recognition[60, 6], pedestrian recognition[42], correlation-based tracking[62].
  - *color-based models-* e.g. skin blob tracking [63, 1].
  - *edge-based models-* e.g. Hausdorff and other edge-based detectors [21, 23].
  - *feature and corner-based models-* e.g. discriminant features [50]
  - *contour-based models-* e.g. condensation-based approaches [24].
  - *heat-based models-* e.g. Infrared-based tracking.
  - **motion-based models-** e.g. frame differencing, background subtraction, flow-based motion tracker [54, 36, 16, 48, 32, 63, 14, 57, 9, 8, 17].
- *probabilistic detection vs.* **deterministic detection-** Some detection approaches determine a discrete set of possible object states ($O_?^t$). Others produce a probabilistic score for a large set of possible object configurations and rely on the correspondence mechanism to determine how many objects are present and their locations.
- *near-field vs.* **far-field** *(articulated vs. gross motion estimation)-* Some approaches are primarily interested in determining the configuration of a single object over time (c_i^t), while others are most interested in determining the proper number of objects and their locations (l_i^t).
- **continuous** *vs. sparse visual evidence-* Some tracking involves tracking objects across frames taken within a short period of time (e.g. tracking a person moving through a room). Other approaches attempt to track across long periods of time (e.g., tracking a person's entrance and exit from a place of work).
- *image-relative vs.* **global state approaches-** Many approaches estimate a part of the state of the object relative to the sensor (O_i^t). This is often sufficient for establishment of correspondence. Other approaches attempt to model a more global state of the object (e.g., location in three dimensional space given multiple cameras). This enables multi-camera tracking and, in some cases, normalization of camera specific properties of objects.
- *single vs.* **multi-sensor-** Many approaches rely on a single sensor whereas other approaches rely on multiple sensors. This is closely related to the question of *image-relative vs. global state.*
- **on-line** *vs. batch-* Some approaches work given the observations up to the current time t, $(Y_i^1, Y_i^2, ...Y_i^t)$, while other approaches rely on having observations over the entire experiment. The following issue is related to this issue.

- **real-time** *vs off-line-* Certain approaches do not lend themselves to real-time processing.
- *one vs.* **many objects-** Some approaches assume a single object is always present. More general approaches assume that object may or may not be present. The most general approaches determine both the number of objects and their state with no more than a loose prior.
- *single object class vs.* **multiple object class-** Some approaches are geared towards only estimating the state of a single type of object, whereas others can handle multiple classes.
- **learned** *vs. manually-specified-* Some aspects of a tracking system can be learned rather than manually specified. For instance, some research efforts expend a large amount of energy in calibrating cameras.
- **automatic** *vs. manual initialization-* Some models must be manually initialized while others claim some robustness.

While this list of descriptive parameters of work in the area of tracking is daunting, many of the factors tend to be correlated. For example, all current articulated (near-field) approaches are limited to a single class of object (e.g. a person). Most rely on a manually-specified model. Many rely on initialization and are not robust to problems such as partial occlusion.

Multi-sensor approaches tend to estimate more global state than single sensor approaches. Online, real-time trackers tend to model less complex aspects of state often using deterministic detection and simple models of observable state.

2.1.2 Previous work in visual tracking

In this section we describe a few major areas of tracking research and discuss their appropriateness for our task.

Articulated modeling

The field of tracking can be split into two major areas of research, near-field (articulated) tracking and far-field tracking. Near-field tracking research includes perception of gesture, articulated body movements, gaze, gait recognition, and other aspects. These methods are primarily concerned with determining the configuration of the objects over time (c_i^t). They generally assume one moving object and consider the location of the object as a nuisance parameter.

We acknowledge the value of these approaches and intend to pursue this line of research after we can successfully model the number of objects, types of objects, their locations, and a rough segmentation of the objects. Also, rather than considering the rest of the object's state as a nuisance parameter, we strongly believe that it will provide essential context for articulated modeling problems. For these reasons, near-field problems will not receive any more attention in this thesis.

Visual detection-based tracking

One major approach to tracking involves using a visual detection mechanism in combination with a method that establishes correspondence. Some examples of detection mechanisms are face detection [60, 43], correlation models [62], skin blobs [63], edge based models [22], contour-based models [24]. Some examples of correspondence methods are simple maximum likelihood tracking (tracking the most likely state from instant to instant), multiple hypothesis tracking, CONDENSATION, or full density approximation.

This type of approach has some advantages. First, the detection mechanism can be used on static input. For instance, a face detection system can search a photo album or the web for faces. Because this is a capability of humans, we have devoted Section 2.5 to describing how our motion-based attention mechanism could be used to bootstrap a static attention mechanism. Second, if the task is only concerned with a particular type of object, this method can reduce the amount of distractors. For instance, a face tracker will only be distracted by things that are detected as faces.

This type of approach also has some limitations that make it less useful in modeling complex environments. It produces a class-specific tracker based on visual appearance. It is possible for some simple models bootstrap an appearance model after a motion based detection as in [62], but this is rarely done in practice as the tracking is often not robust to changes in object appearance. In general the detection mechanism is trained with a large supervised set of data.

We have looked at bootstrapping this type of system as a secondary attention mechanism. Because of the disadvantages of this approach we did not consider it for our base tracking system. Our primary motivation is the vast variability of tracked objects we have observed in our tracking experiments (see Table 2.1). While it is hard to imagine the usefulness of detecting all of the objects listed in Table 2.1, it is obvious that limiting yourself to a single or small set of object types would fundamentally limit your system's understanding of the environment.

Some general models attempt to model the world with layers or sprites citewangadelson94. These are purported to be very general solutions. Jojic and Frey have recently created a system that can model layers given video input with considerable occlusions [29]. Unfortunately, the computational requirements and the robustness of these approaches make them inappropriate for our needs.

Motion-based tracking

Our primary goal in tracking is to have a robust, general tracking system that can track the positions and appearances of many objects of many different types in many environments over extended periods of time. Rather than trying to filter out all the moving objects that might be considered distractors in Table 2.1, we track them and rely on later processing to model them.

We also wanted to produce a real-time tracking system that is computationally feasible on the hardware available when this work began. In short, we want a far-field, multiple object, multiple object class, on-line, real-time tracking system. As

a result of these requirements we determined our base tracking system should be a motion-based, continuous tracker, which relies on deterministic detection. This made background estimation on static cameras an obvious choice.

Being far-field, motion-based, and continuous, it was not tuned for particular type of objects and relied on reasonably universal object dynamics. Once it became apparent that we could create a robust reliable tracking system for sparsely populated environments, secondary goals arose. First, we looked at bootstrapping a static attention mechanism based on the motion-based mechanism. In as much as possible, we wanted to unify the state estimates across multiple sensors through time.

2.1.3 Previous work in background maintenance

The terse description of the field of visual tracking given above does little to describe the diverse approaches to background estimation. In this section, we will detail previous work in background estimation.

Most researchers have abandoned non-adaptive methods of backgrounding because of the need for manual initialization. Without re-initialization, errors in the background accumulate over time, making this method useful only in highly-supervised, short-term tracking applications without significant changes in the scene. It is possible to use a maximum interframe difference[36], but this leaves "ghosts" where the object was and leaves large regions of the object undetected unless the object undergoes significant motion each frame.

Most backgrounding methods involve continuously estimating a statistical model of the variation for each pixel. A common method of adaptive backgrounding is averaging the images over time, creating a background approximation which is similar to the current static scene except where motion occurs. While this is effective in situations where objects move continuously and the background is visible a significant portion of the time, it is not robust to scenes with many moving objects particularly if they move slowly. It also cannot handle bimodal backgrounds, recovers slowly when the background is uncovered, and has a single, predetermined threshold for the entire scene. One interesting attempt to meet these difficulties is W^4[16], which combined its estimates of the minimum value, maximum value, and maximum interframe difference per pixel.

Ivanov[25] used disparity verification to determine moving regions in a scene. This showed invariance to lighting variations but involved a costly, off-line initialization. Its primary application is for geometrically static backgrounds. Recently, an eigenvector approximation of the entire image was used to model the background in outdoor scenes[41], but the difficulties of on-line estimation have limited its applicability.

Changes in scene lighting can cause problems for many backgrounding methods. Ridder et al.[48] modeled each pixel with a Kalman Filter which made their system more robust to lighting changes in the scene. While this method does have a pixel-wise automatic threshold, it still recovers slowly and does not handle bimodal backgrounds well. Koller et al.[32] have successfully integrated this method in an automatic traffic monitoring application.

Pfinder[63] uses a multi-class statistical model for the foreground objects, but the

MOVING OBJECTS	OR CHANGES TRACKED BY OUR SYSTEM
OBJECTS	
vehicles	cars, trucks, semis, buses, vans, mini-vans, pickups, motorcycles, trains, planes, headlights/taillights/ running lights(at night)
people	individuals adult children people with strollers groups of people rollerbladers bikers flash lights(at night)
scene elements	branches dust on a road grass, bushes, flowers, etc. sprinklers flags and flag shadows traffic lights fire fireworks garage doors doors windows opening or closing inter-reflections shades being drawn business fluorescent lights construction cranes(6 or more blocks away) traffic gates construction signs arrows, blinkers, etc. advisory signs
water	bodies of drainage steam
misc objects	blowing trash bags & newspapers manipulated objects sticks thrown bats, balls, frisbees, etc. trash cans/dumpsters
animals	birds (pigeons, seagulls, swans) squirrels dogs cats other(skunk, raccoon, horse, etc.)
LIGHTING RELATED	
shadows	of terrestrial moving objects, of stationary objects, of planes, blimps, etc.
reflections	windows, wet pavement, water surface, vehicle windows or hoods
direct-lighting changes	partially cloudy, daily lighting cycles, area lighting going on or off, vehicle lights
SCENE CHANGES	
	new buildings/structures, changes resulting from severe weather, vandalism, wet pavement and tire trails, weather, clouds, snow, hail, rain, seasonal(longer-term), flowers, falling leaves
CAMERA RELATED	
	camera damage/degradation, dirty lenses, fly on lens
INDOORS	
movable objects	leds updatable information displays TVs projectors monitors tickers static displays which flicker(TV, Monitors, etc.) lights on/off fish tanks(fish, bubbles, etc.) manipulated objects! papers food chairs staplers, phones, etc. doors shades

Table 2.1: This table shows some of the moving objects that have been tracked by our tracking system. While some may be more important than others in particular tasks, they are all potentially important in modeling an active environment. The considerable variability justifies a general tracking approach.

background model is a single Gaussian per pixel. After an initialization period where the room is empty, the system reports good results. There have been no reports on the success of this tracker in outdoor scenes or in situations with multiple moving objects.

Friedman and Russell[14] implemented a pixel-wise EM framework for detection of vehicles that bears the most similarity to our work. Their method attempts to explicitly classify the pixel values into three separate, predetermined distributions corresponding to the road color, the shadow color, and colors corresponding to vehicles. Their attempt to mediate the effect of shadows appears to be somewhat successful, but it is not clear what behavior their system would exhibit for pixels which did not contain these three distributions. For example, pixels may present a single background color or multiple background colors resulting from repetitive motions, shadows, or reflectances.

2.1.4 Our approach to motion tracking

Rather than explicitly modeling the values of all the pixels as one particular type of distribution, we simply model the values of a particular pixel as a mixture of Gaussians. Based on the persistence and the variance of each of the Gaussians of the mixture, we determine which Gaussians are most likely to represent pixel values from static objects. Pixel values that do not fit the background distributions are considered foreground until there is a Gaussian that includes them with sufficient, consistent evidence supporting it to convert it to a new background mixture component.

Our system adapts to deal robustly with lighting changes, repetitive motions of scene elements, tracking through cluttered regions, slow-moving objects, and introducing or removing objects from the scene. Slowly moving objects take longer to be incorporated into the background, because their color has a larger variance than the background. Also, repetitive variations are learned, and a model for the background distribution is generally maintained even if it is temporarily replaced by another distribution which leads to faster recovery when objects are removed.

Our backgrounding method contains two significant parameters – α, the learning constant and T, the proportion of the data that should be accounted for by the background. Without any alteration of parameters, our system has been used in an indoors, human-computer interface application and has been continuously monitoring outdoor scenes since October 1997.

Since the publications of our results, many interesting systems have been developed. Toyama et al. [57] developed a system called "Wallflower" that had three components: a pixel-level component, a region-level component, and a frame-level component. Their system compared favorably to nine other systems on seven different problems.

Ellis and Xu [9] used a mixture of Gaussians in color chromaticity rather than (R, G, B) to attempt to be more robust to lighting changes. Elgammal et al. [8] use a full density approximation rather than a parametric mixture of Gaussians. Haritaoglu et al. [17] integrated shape and motion cues to improve estimation of their model.

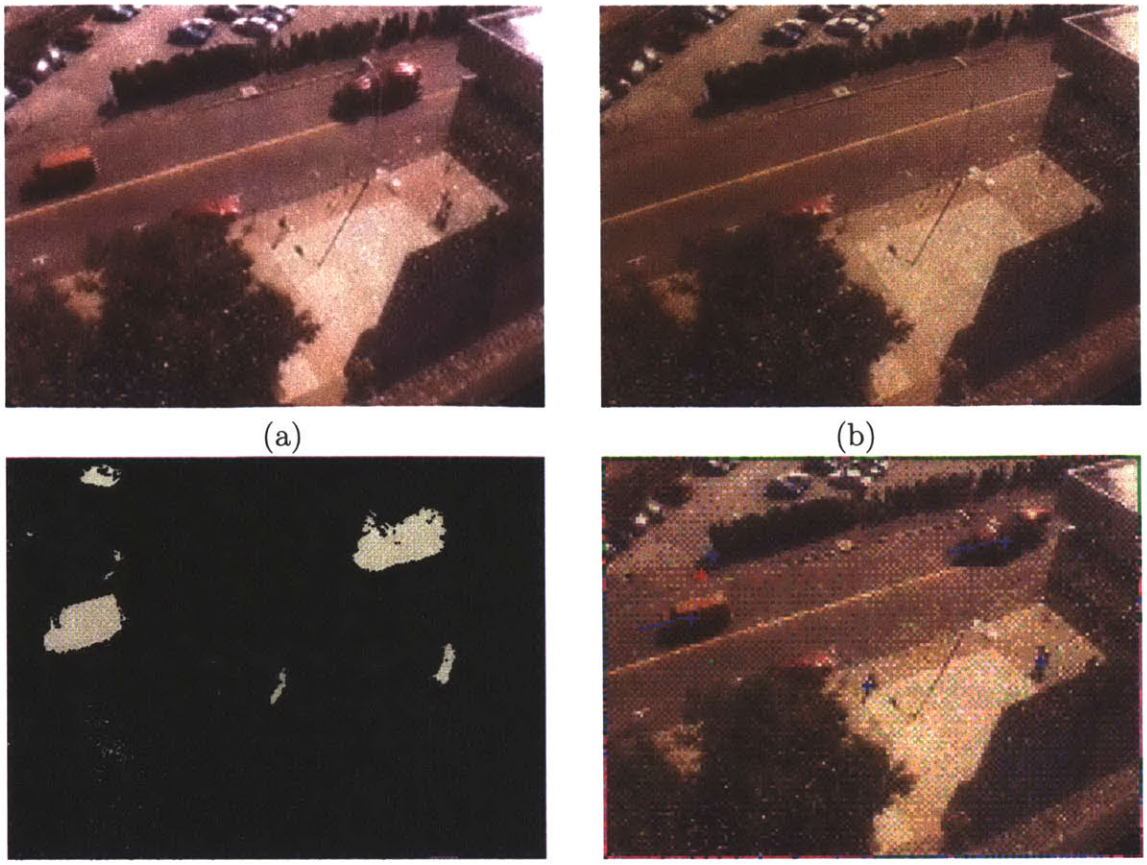


Figure 2-1: *The execution of the program. (a) the current image, (b) an image composed of the means of the most probable Gaussians in the background model, (c) the foreground pixels, (d) the current image with tracking information superimposed. Note: the shadows are foreground in this case because the surface was not covered by shadows a significant portion of the time. Hence, no Gaussian representing those pixel values was significant enough to be considered background.*

2.2 Adaptive background maintenance for motion tracking

The underlying hypothesis of background maintenance is that the world is composed of static objects and active objects. The goal of adaptive background maintenance is to determine whether each pixel in an image sequence is measuring the radiance of the surface of a static object or an active object. This is a binary classification problem. Unfortunately, no explicit model of the background or foreground objects can be estimated except in cases where all active objects are removed for a training period and all other factors remain the same.

Figure 2-1 shows an example of foreground vs. background segmentation produced by our tracking system. The regions corresponding to the moving vehicles and pedestrians are effectively classified as foreground (white) pixels. The regions that

are not moving are primarily classified as background (black) pixels. There are some errors, but they are not localized and can be filtered.

2.2.1 Adaptive Background Mixture Model (ABMM) Overview

Because of the computational complexity of representing pixel-wise dependencies, almost all current background estimation techniques assume independent models for each pixel. This is a limitation that may be overcome in the next few years. Our work currently makes this assumption.

Most adaptive background models attempt to represent the background distribution and consider pixel values that do not fit that model to be foreground pixels. This assumes that either the foreground pixel values are not significant in the model estimation or are factored out of the estimation. For example, the single Gaussian (averaging) model assumes the background is unimodal and the values in the foreground will not significantly affect this estimation at the chosen time scale.

On the other hand, our mixture models represent both the foreground and background pixel values with a single mixture of Gaussian distributions. Whether a pixel value is considered foreground or background is dependent on the characteristics of the distribution that represents that value. In effect we are segmenting the observations based on multiple models and then determining which set of models are likely to result from background surfaces.

The following sections describe our approach to background modeling. The first subsection describes the observations made at each pixel–the "pixel process". The following subsection describes online estimation of the parameters of the mixture of Gaussian estimate from pixel values. The following subsection describes how to determine which components of the mixture of Gaussians are most likely to result from static objects. This allows the pixel values to be labeled as foreground or background. The following section describes how to efficiently group the pixels into connected regions. Appendix A has a pseudo-code description of both of these processes.

2.2.2 The "pixel process"

Each pixel value is a noisy measurement of the surface radiance of the first object that its optical ray intersects in the scene. The values of a particular pixel over time can be considered a "pixel process", i.e. a time series of scalars for grayvalues or vectors for color pixel values. At any time, t, what is known about a particular pixel $I(x_0, y_0)$ is its history

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \tag{2.1}$$

where I is the image sequence and X_i is the pixel value.

If each pixel resulted from a single surface under fixed lighting, a single Gaussian would be sufficient to model the pixel value while accounting for acquisition noise. If only lighting changed over time, a single, adaptive Gaussian per pixel would be sufficient. Surfaces of active objects can occlude the background in a particular pixel

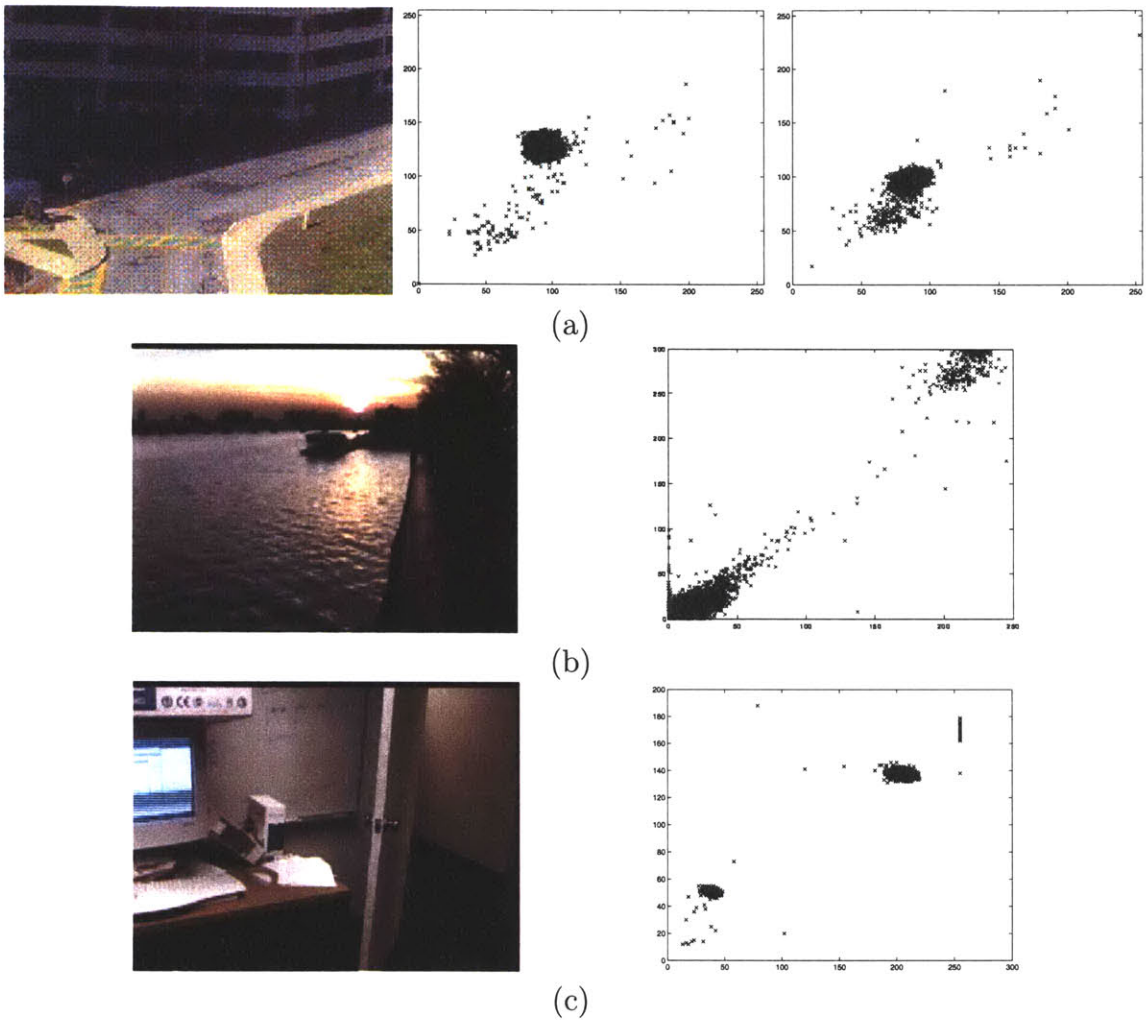


Figure 2-2: *This figure contains images and scatter plots of the red and green values of a single pixel from the image over time. It illustrates some of the difficulties involved in real environments. (a) shows two scatter plots from the same pixel taken 2 minutes apart. The two distributions show different variances and means. (b) shows a bi-modal distribution of a pixel's values resulting from specularities on the surface of water. (c) shows another bi-modality resulting from monitor flicker.*

and the lighting conditions change. Thus, multiple, adaptive Gaussians are required. We use an adaptive mixture of Gaussians to approximate this process.

Some "pixel processes" are shown by the (R,G) scatter plots in Figure 2-2, which illustrate the need for adaptive systems with automatic thresholds. Figure 2-2(b) and (c) also highlight a need for a multi-modal representation. In each case, the ideal distribution of values should be a tight, Gaussian-like cluster around some point. The fact that the cluster can shift dramatically over a period of a few minutes or that two or more processes at the same pixel can result in several distinctive clusters illustrates the need for an adaptive, multi-modal representation.

2.2.3 Online mixture model

At any point in time t, only the current and past values of each pixel are available. We model the recent history of each pixel, $\{X_1, ..., X_t\}$, with an approximation to a mixture of K Gaussian distributions. The probability of observing the current pixel value given our model is

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{2.2}$$

where K is the number of distributions, $\omega_{i,t}$ is an estimate of the weight (the portion of the data accounted for by this Gaussian) of the i^{th} Gaussian in the mixture at time t, $\mu_{i,t}$ and $\Sigma_{i,t}$ are the mean value and covariance matrix of the i^{th} Gaussian in the mixture at time t, and where η is a Gaussian probability density function

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)} \tag{2.3}$$

K is determined by the available memory and computational power. Currently, values ranging from 3 to 5 are used. Also, for computational reasons, the covariance matrix is assumed to be of the form:

$$\Sigma_{k,t} = \sigma_k^2 \mathbf{I} \tag{2.4}$$

This assumes that the red, green, and blue pixel values are independent and have the same variances. While the noise is certainly not spherical, this assumption allows us to avoid a costly matrix inversion at the expense of reduced accuracy. Using a diagonal covariance would allow a Gaussian to represent that a particular channel showed more variation. Using a full covariance matrix would allow each Gaussian to model the its local variation with more accuracy. This would be particularly helpful in modeling of variation due to lighting, which varies significantly across the color space.

Thus, the distribution of recently observed values of each pixel in the scene is characterized by a mixture of Gaussians. Each new pixel value will be represented by one of the major components of the mixture model and used to update the parameters of that component of the mixture.

If the pixel process could be considered a stationary process, a standard method for determining the parameters that maximize the likelihood of the observed data is *expectation maximization*[7]. Because there is a mixture model for every pixel in the image, implementing an exact EM algorithm on a window of recent data would be costly. Also, lighting changes and the introduction or removal of static objects suggest a decreased dependence on observations further in the past. These two factors led us to use the following on-line best-first K-means approximation to a Gaussian mixture model.

Every new pixel value, X_t, is checked against the existing K Gaussian distributions (starting with the most likely background Gaussians) until the first match is found. A match is defined as a pixel value within 2.5 standard deviations of a distribution[1]. This threshold can be perturbed with little effect on performance. This is effectively a per pixel/per distribution threshold. This is extremely useful when different regions have different lighting (see Figure 2-2(a)), because objects which appear in shaded regions do not generally exhibit as much noise as objects in well-lit regions. A uniform threshold often results in objects not being detected after they enter shaded regions.

If none of the K distributions match the current pixel value, the least probable distribution is replaced with a distribution with the current pixel value as its mean value, an initially high variance, and low prior weight.

The prior weights of the K distributions at time t are adjusted as follows

$$\omega_{k,t} = \frac{(1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})}{Z} \tag{2.5}$$

where Z is a normalization constant, α is the learning rate[2], and $M_{k,t}$ is 1 for the model which matched and 0 for the remaining models. $1/\alpha$ defines the time constant that determines the speed at which the distribution's parameters change. $\omega_{k,t}$ is effectively a causal low-pass filtered average of the (thresholded) posterior probability that pixel values have matched model k given observations from time 1 through t. This is equivalent to the expectation of this value with an exponential window on the past values.

The mean (μ) and variance (σ) parameters for unmatched distributions remain the same. The parameters of the distribution which matches the new observation are updated as follows

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t \tag{2.6}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{2.7}$$

where

$$\rho = \alpha\eta(X_t; \mu_k, \sigma_k) \tag{2.8}$$

[1]Depending on the kurtosis of the noise, some percentage of the data points generated by a Gaussian will not "match". The resulting random noise in the foreground image is easily ignored by neglecting connected components containing only a few pixels.

[2]While this rule is easily interpreted an an interpolation between two points, it is often shown in the equivalent form: $\omega_{k,t} = \omega_{k,t-1} + \alpha(M_{k,t} - \omega_{k,t-1})$

is the learning factor for adapting current distributions[3]. This is effectively the same type of causal low-pass filter as mentioned above, except that only the data which matches the distribution is included in its history for the purposes of estimation.

One of the significant advantages of this method is that when pixel values resulting from a new object are allowed to become part of the background, the existing model of the background is not destroyed. The original background color remains in the mixture until it becomes the K^{th} most probable mixture component and a new color is observed. Therefore, if an object is stationary just long enough to become part of the background and then it moves, the distribution describing the previous background still exists with the same μ and σ^2 but a lower ω, and will be quickly re-incorporated into the background.

2.2.4 Background model estimation

As the parameters of the mixture model of each pixel change, we would like to determine which components of the Gaussian mixture model are most likely to represent observations of static surfaces. Heuristically, we are interested in the Gaussian distributions which have the most supporting evidence and exhibit the least variance.

To understand this choice, consider the accumulation of supporting evidence and the relatively low variance for the "background" distributions when a static, persistent object is visible. In contrast, when a new object occludes the background object, it will not, in general, match one of the existing distributions, which will result in either the creation of a new distribution or the increase in the variance of an existing distribution. Also, the variance of the moving object is expected to remain larger than a background pixel until the moving object stops. For example, a loitering pedestrian not remain as stationary as most dropped objects. To model this, we need a method for deciding what portion of the mixture model best represents background processes.

First, the Gaussians are ordered by the value of ω/σ. This value increases both as a distribution gains more evidence and as the variance decreases. While background surfaces that are rarely present or have high variance will tend to be more quickly replaced than those that have more evidence and exhibit less variation. After re-estimating the parameters of the mixture, it is sufficient to sort from the matched distribution towards the most probable background distribution, because only the matched models' relative value will have changed. This ordering of the model is effectively an ordered, open-ended list, where the most likely background distributions remain on top and the less probable transient background distributions gravitate towards the bottom and are eventually replaced by new distributions. Given enough computational power, this approximation would not be necessary.

Then the first B distributions are chosen as the background model, where

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > T \right) \tag{2.9}$$

[3]In high dimensional spaces with full covariance matrices, it is sometimes advantageous to use a constant ρ $(= \alpha)$ to reduce computation and provide faster Gaussian tracking.

where T is a measure of the minimum portion of the data that should be accounted for by the background. This takes the "best" distributions until a certain portion, T, of the recent data has been accounted for. If a small value for T is chosen, the background model is usually unimodal. If this is the case, using only the most probable distribution will save processing.

If T is higher, a multi-modal distribution caused by a repetitive background motion (e.g. leaves on a tree, a flag in the wind, a construction flasher, etc.) could result in more than one color being included in the background model. This results in a transparency effect which allows the background to accept two or more separate colors.

2.3 Evaluating our method

Background estimation works very well in many situations, but fails in certain situations where assumptions are violated. Here we outline a set of these problems (similar to Toyama et al. [57]).

- **Regular object occlusions**- The most basic problem is occlusions of the background by transient objects. The background estimation procedure must be robust to both short-term occlusions and long-term occlusions.
- **Object appearance not significantly different from background**- The most basic assumption is that the pixel values resulting from active objects are separable from the pixel values of static objects. If this is not true, little can be done using local methods. Our system assumes this is not the case.
- **Object type transition**- Active objects can become static objects and vice versa. This problem is also referred to as the sleeping object and waking object problem. An example of this problem is a car which moves into a parking lot, parks, and doesn't move until the next day. If the object is not incorporated into the background, it will disrupt the tracking of any object that moves in front of it. When that object moves the next day, it re-exposes the cement (which most models have "forgotten") causing a ghost. This is related to the problem of bootstrapping a model without a training period. This can occur over a short period (e.g., vandalism, dropped objects, retrieved objects, etc.) or over a longer period of time (e.g., new structures, fallen trees, etc.).
- **"False" moving objects**- There are many types of objects or motion artifacts that exhibit the characteristics of moving objects that we may not want to track. In most cases, our system tracks these artifacts and relies on later processing to filter them out. Table 2.1 lists many of these objects. Here we list a few categories.
  - shadows- The most common "false" object is a shadow. Our system tracks the shadow as part of the object except in cases where the shadow appears a significant amount of the time.

- other object-related artifacts- Shadows are only a small portion of the object-related artifacts we have observed. On rainy days, there are reflections off roads. On snowy days, tracked objects leave visible tracks. Cars have headlights and fog lights.
- repetitive motions- Branches blowing in the wind, dust on a road, sprinklers, steam, and flags can produce motion artifacts in such quantities that it dwarfs the amount of data that comes from other objects.
- information displays and indicators- Displays and indicators are common in both indoor and outdoor environments. Monitors, LEDs, and TVs are common indoors. Construction indicators and traffic lights are common outdoors.

- **Environmental conditions**- Lighting variation and environmental factors cause many problems, particularly in outdoor scenes. Fast lighting variations occur from lights being turned on or off or lighting strikes. Slow lighting variations can occur as a result of moving cloud cover or doors or windows being opened or closed. Long-term changes also result from changes in weather or season.

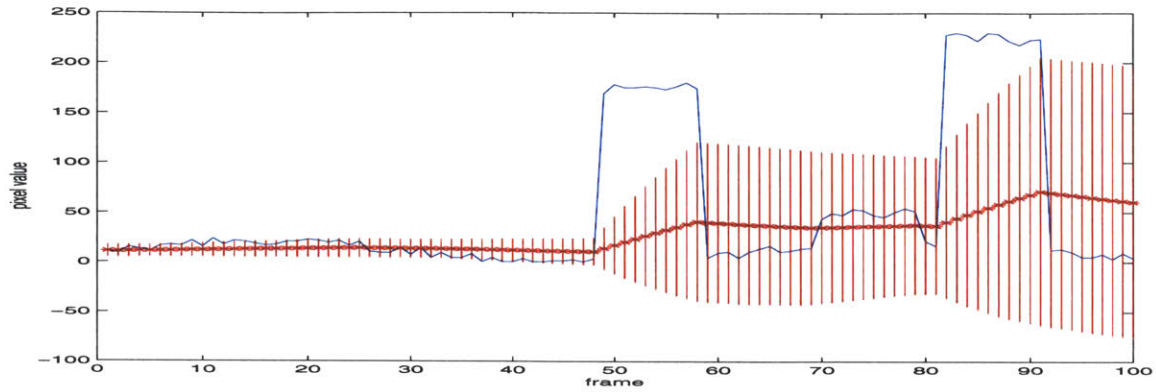
2.3.1 A simple example

While it is difficult to evaluate our method of background estimation empirically, we will show its effectiveness on a simple example. Given a single pixel's greyscale value over a short period with multiple occlusions caused by foreground objects, we show two approaches to background estimation in Figure 2-3. While this does little to elucidate the complexity of this problem, it is helpful in building intuitions for understanding our approach.

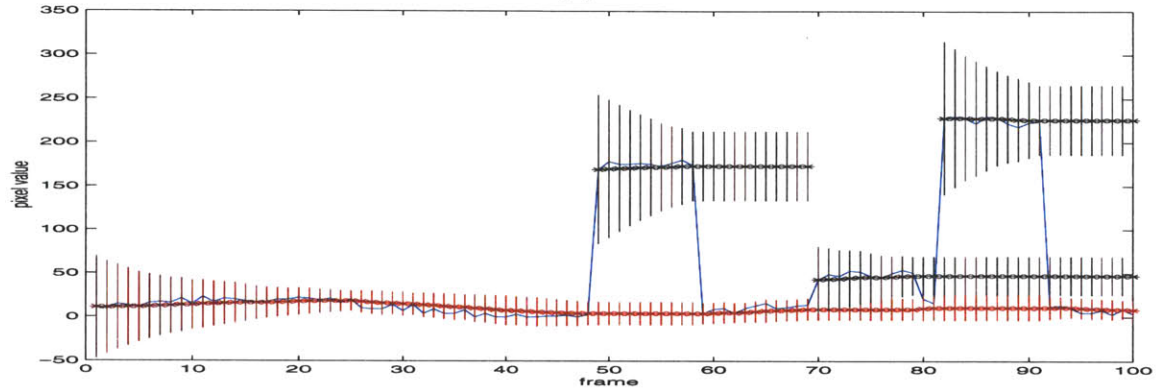
The first approach is a unimodal background model. The learning thresholds were too slow to adapt to the slow changes in the background colors, yet enabled the background model to be extremely corrupted by a short occlusion. The recovery time for the short occlusion events are significant. Our approach begins with one model whose weight (darkness) increases and variance (error bar) decreases as more evidence is attributed to that component. Three short occlusion events initialize three Gaussian color models. Each of the three additional models follows the same pattern but never becomes significant enough to be a background component. Most importantly, the background model that took time and evidence to be established is not corrupted by the new observations.

2.4 Connected components

The method described above allows us to identify foreground pixels in each new frame while updating the description of each pixel's process. These labeled foreground pixels can then be segmented into regions by an efficient two-pass, connected components algorithm [20]. Details of this method are covered in Appendix A



(a)



(b)

Figure 2-3: *This figure shows a simple example of background estimation over time. (a) is a unimodal Gaussian model. The mean of the Gaussian is plotted as well as variance error bars. (b) is our multi-model Gaussian model. The relative weights of the different models are shown in variation from black (largest weight) to white (no weight). This example shows the creation of three different models to account for three different occlusion events. The variance tightens and the weight increases when the values are regular. The variance loosens when the Gaussian is tracking the values because the mean estimate is not accurate.*

Because this procedure is effective in determining the whole moving object, moving regions can be characterized not only by their position, but size, moments, and other shape information. Not only can these characteristics be useful for later processing and classification, but they can aid in the tracking process.

2.5 Bootstrapping static attention from an active environment

Using an adaptive background estimation technique limits the applicability of this body of work to situations where static cameras are observing environments that contain active objects that move independently and do not regularly occlude each other. The details of an existing system to bootstrap a pedestrian detection system from a tracking system in an environment with only pedestrians are given in Section 4.6.5. This capability would allow a system to be introduced into more and more complex environments and still be able to function.

Chapter 3

Establishing object correspondence

The previous chapter discussed a method for determining a discrete set of possible moving object observations in each video frame of each camera in the system. This enables us to determine that an object with a certain appearance was present at a particular location in sensor j at time t. On its own, this information can tell little about the objects and activities of a particular environment beyond camera-specific models of where objects tend to be.

Fortunately, each of these observations were caused by an object in the world, and many may have been caused by the same object over time or in multiple sensors. By taking advantage of this fact, we can reduce the redundancy in our data by grouping our observations into sets of observations of the same object, or multiple observation sets (MOSs). A perfect correspondence system could determine the minimum number of MOSs containing only observations from a single object regardless of which day the object was recorded. The goal in this chapter is to determine a set of MOSs that contain only object observations corresponding to a *single* object rather than attempting to determine MOSs that contain *all* the object observations corresponding to a particular object (but potentially grouping observations of multiple objects together). Chapter 4 will illustrate that MOSs contain information that enables effective clustering based on identity.

In this chapter we discuss many problems in establishing object correspondence. Section 3.1 introduces object correspondence and describes an *ideal* object correspondence system. The following three sections describe the three basic problems in object correspondence. The three types of object correspondence problems are instantaneous object correspondence, continuous object correspondence, and discontinuous object correspondence. Figure 3-1 shows these three essential problems in establishing object correspondence on a timeline. Not all aspects of each correspondence problem are addressed directly in this thesis, but those not covered are briefly reviewed for the benefit of the reader. Section 3.5 describes how our models can be used to aid in normalization of relatively constant properties of objects.

Below we detail the three basic problems in object correspondence.

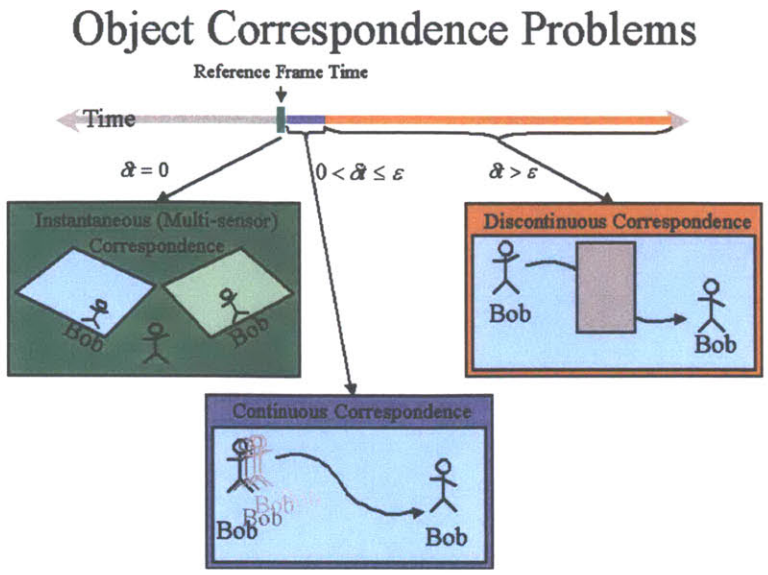


Figure 3-1: *This figure shows the three essential problems in establishing object correspondence. Instantaneous object correspondence is correspondence between pairs of observations of the same object at the same time in different sensors. Continuous object correspondence is correspondence over time. Discontinuous object correspondence is correspondence after missing observations caused by distraction or objects leaving the environment.*

Instantaneous (multiple camera) object correspondence

When the view frustra of two or more cameras overlap and an object is present in the region of overlap, it may result in multiple object observations. Given enough redundant observations, it is possible to build models of visibility and correspondence for the areas of overlap. Using these models it is possible to determine that multiple observations resulted from the same object in the world. Although there is a large, related field in camera calibration and establishing correspondence in images from visual features, correspondence points from tracked objects have a very different character and deserve special attention.

We will discuss the three possible relationships between pairs of cameras as well as a mechanism for determining which case is present in a particular situation. The cases are:

- *detected objects in the region of visual overlap occupy a significant portion of a three dimensional subspace-* When there are enough reliable object detections between two cameras and those detections are not approximately planar, external camera calibration is possible. Given the corresponding object observations and the internal camera parameters describing the optics of the camera, it is possible to estimate the external camera parameters including relative position and relative angle. Multiple cameras watching birds and people in swimming pools are obvious examples where objects occupy a significant portion of a three

dimensional subspace. Knowing the camera geometry and a point in one image corresponding to the position of an object in the environment, the corresponding point in the second image is restricted to lie on a line. This epipolar constraint is useful for establishing correspondence of observation sequences but underconstrained for instantaneous correspondence because the position of an object in the first image could potentially match to many the position of multiple objects in the second image.

- *detected objects lie on one or more low dimensional subspaces*- Because of gravity, most objects lie near surfaces in the world. Taking advantage of this regularity allows one to determine a one-to-one mapping of points in one image to points in the other image in many cases. In fact, if all objects are detected near a *single* plane, external camera calibration is not possible. Also, external camera calibration can be made less robust due to various shortcomings in the estimation of the projection of the centroid of the object due to noise, occlusions, shadows, reflections, and other factors. We cover two methods for modeling this case:

  - single homography- Often tracked objects move on a single ground plane in the area of overlap. It is possible to robustly estimate the homography that maps the centroid of an object in one camera to the centroid of the same object in the other camera. We discuss previous work in this area done by our group as well as adding visibility constraints and anomaly detection.
  - multiple homographies- This is an additional layer to the above method. There are many situations where a single homography is insufficient. This includes multiple planes of correspondence, occlusions, shadows, reflections, and objects with different centroid heights.

- *no objects detected in region of visual overlap*- Because we do not specify the camera configuration to our system, it is important to understand when there *is* and when there *is not* visual overlap. This is a very important case to understand, because, in many situations, the majority of pairs of cameras will not overlap or will not have active objects detected in a region of visual overlap.

Continuous object correspondence

In the case of an object moving in full view, the object can be tracked from frame to frame. In many cases, very simple dynamics can be used to effectively track objects from frame to frame. When there are multiple object observations in similar locations with similar characteristics, this problem becomes more difficult. We will discuss a variety of approaches to this problem with particular attention to the one used in our system, multiple hypothesis tracking (MHT). We will also discuss some results in different environments.

Discontinuous object correspondence

Often observations of an object are not available for a period of time and it is necessary to attempt to re-establish correspondence. Objects can be "lost" due to interactions with other moving objects, visual effects that disrupt the functioning of the attention mechanism, or short occlusions. There are many characteristics of objects that can be employed to re-establish correspondence including:

- dynamics models- Based on the movement of the object it is often possible to estimate a likelihood that an newly tracked object corresponds to a previously tracked object [24].
- appearance models- Based on the appearance of the tracked object it is also possible to help re-establish correspondence [16].
- behavioral models- Based on the dynamic properties (e.g., a fast walking) of the object or the intended behavior of an object, it is possible to infer that a newly tracked object corresponds to a previously tracked object even after significant periods of missing observations.

Continuous and discontinuous object correspondence are most often associated with tracking in single sensors. Correspondence across sensors is useful in multiple object tracking or tracking in extended environments.

Normalization of regular properties

We will also discuss methods for normalizing certain characteristics of a single camera. In some cases the normalization corresponds to finding an orthogonal projection of the data. Properties that can sometimes be normalized are velocity, size, and height. For instance, after normalization the estimated size of an object far from the camera and near to the camera should be approximately equal. This decreases the complexity of modeling object and activity types. Also, with as little as one labeled velocity, size, or height, it is possible to know the exact velocity, size, or height of objects in the environment. This will increase the amount of transfer of models from one site to another. In the case of Euclidean camera calibration, this normalization is unnecessary, but in less ideal situations this technique is useful.

3.1 Ideal object correspondence

Every observation, o_x, is given a unique index, x. To define an MOS, we introduce a labeling function, $U(x)$, which specifies a unique identifier for each observation that corresponds to a particular object. If the true object labels U_x were known, an ideal solution would be $U(x) = U_x$. If $U(x) = x$, each observation would be from a unique object. If $U(x) = const$, every observation would be from the same object. Since these labels are not specific to particular objects (e.g., 'bob', 'john', etc.), any permutation of the unique labels would be equivalent.

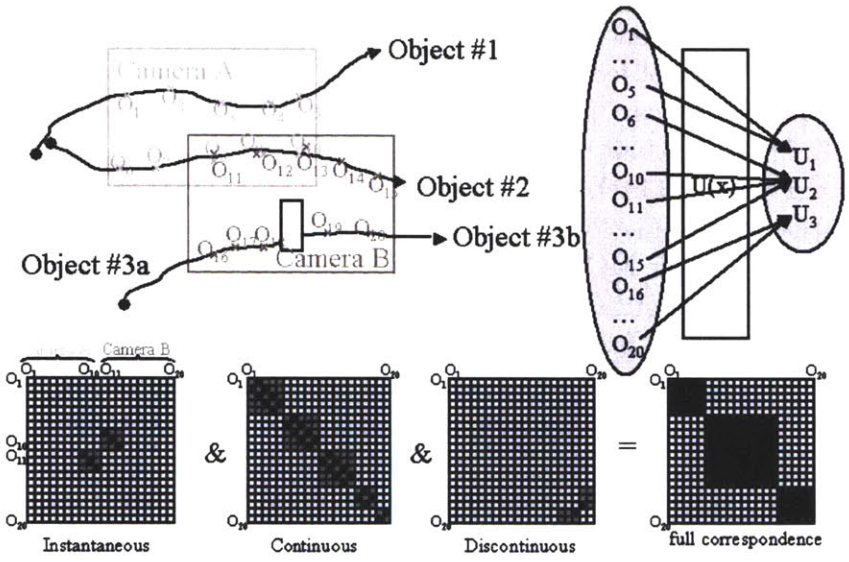


Figure 3-2: *This figure shows three objects tracked through two sensors and the resulting correspondence matrix. Object 1 is tracked only in camera A. Object 2 is tracked in camera A and camera B. Object 3 is tracked through occlusion in camera B. Instantaneous, continuous, and discontinuous correspondences are highlighted in the example correspondence matrices below. In the upper right, an ideal labeling function has assigned* $[O_1, ..., O_5]$ *to one unique identifier,* $[O_6, ..., O_{15}]$ *to a second unique identifier, and* $[O_{16}, ..., O_{20}]$ *to a third unique identifier. Regardless of the values of* (U_1, U_2, U_3)*, the full correspondence matrix in the lower right will result.*

A multiple observation set (MOS), M^u is a set of observations $\{O_{x_0}, O_{x_1}, ..., O_{x_T}\}$ whose unique label is the same value (u). E.g.,

$$U(x_0) = U(x_1) = ... = U(x_T) = u \tag{3.1}$$

The observation correspondences for the complete set of D object observations can be summarized in a DxD object correspondence matrix, K, where

$$K_{i,j} = \begin{cases} 1 & \text{if } U(i) = U(j) \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

An ideal object correspondence system would establish N multiple observation sets (MOSs) that correspond to observations of each object. Each set would contain all observations of that object regardless of when it occurred or which sensor made the observation. Every car, every person, every piece of trash would have a single corresponding MOS. Nothing about the true identity of the MOS would be available, but every observation in each MOS would correspond to a single object. Whenever that object appeared in any sensor, the system would assign the corresponding object observation to the proper MOS by assigning it the proper label.

Without perfect tracking in a completely observable system or user supervision in a restricted domain (e.g., face recognition with a small domain), there is little hope of ever achieving this ideal goal. On the other hand, the objects in the world obey certain regularities. For instance, only one object can exist at the same point in space at the same time. Also, the state of an object must vary continuously through time. This chapter discusses how to exploit these and other regularities.

Figure 3-1 shows the three essential problems in establishing object correspondence. A simple example of correspondence labeling is shown in figure 3-2.

Section 3.2 discusses the problem of instantaneous camera calibration including the three possible cases of overlap between cameras and how to differentiate between them. Section 3.3 discusses the problem of continuous object correspondence–tracking objects given continuous observations of the objects. Section 3.4 discusses problems related to re-establishing correspondence when the observations are not continuous.

3.2 Instantaneous (Multiple camera) Object Correspondence

Instantaneous object correspondence can be simply stated as determining which pairs of observations taken from two cameras at the same time result from the same object. After some discussion of related research, we cover three different cases of camera overlap.

Related research

Tracking in extended scenes using distributed cameras is a difficult and interesting problem. Many individuals have attempted to model relationships between tracked

objects in multiple cameras without significant overlap [30] and cameras with significant overlap [35]. This chapter is concerned with scenes with significant overlap between cameras in which tracked objects are present.

Javed et al. [26] modeled the projections of the location of the feet of pedestrians as they leave one camera in the other camera. In previous work at the MIT AI Lab[35], we have shown that for scenes with a single plane of correspondence we can estimate the corresponding homography robustly despite the large number of false correspondences resulting from multiple objects being tracked at the same time. We have also shown that time alignment can be robustly estimated in such cases where exact time synchronization is not available. In later work[35], we showed that these homographies are robust enough to estimate camera positions when three pairs of cameras are present.

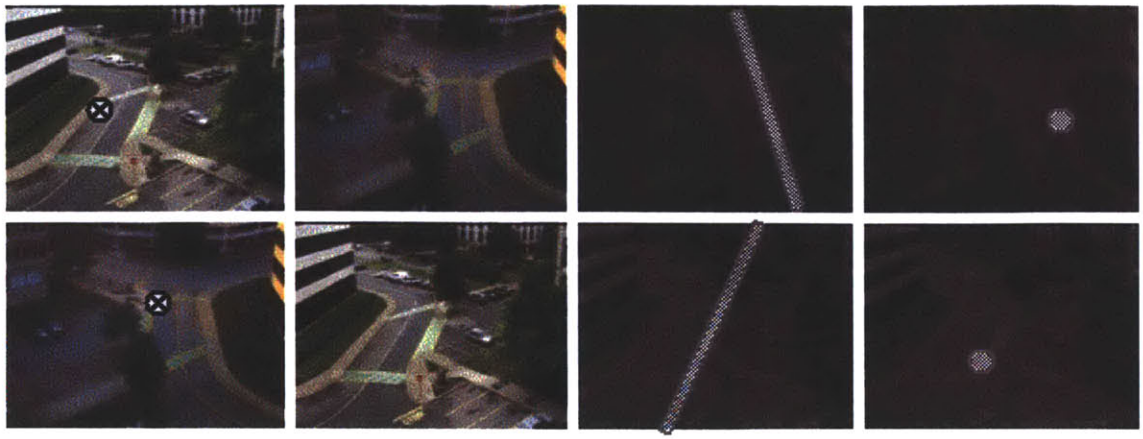
The field of reconstruction has also applied statistical estimation to finding multiple planes of correspondence [65, 4], but the characteristics of correspondence points of tracked objects are fundamentally different than visual correspondence features. When trying to find multiple planes of correspondence in a pair of cameras without rough camera calibration or previous knowledge, the search is vast [37] With knowledge of the orientation of the plane of correspondence [65], the problem becomes easier. With moving cameras [65] adjacent frames are taken from very similar positions and the problem is a local search.

In the following sections, we investigate the characteristics of tracked object centroids in multiple cameras as correspondences. The characteristics of tracked object correspondences allow greatly reduced number of false correspondences and more robust estimation of reliable sets of correspondence pairs. This is followed by a discussion of the different types of correspondence models.

Overview

The goal of instantaneous object correspondence is to determine whether it is more likely that a pair of observations from two cameras resulted from the same object or two different objects. Given a pair of observations from two cameras, (O_A, O_B), it is possible to model this likelihood ratio as

$$
\begin{aligned}
f(O_A, O_B) &= \frac{p(O_A, O_B | l(A) = l(B))}{p(O_A, O_B | l(A) \neq l(B))} && (3.3)\\
&= \frac{p(O_A, O_B | l(A) = l(B))}{p(O_A)p(O_B)} && (3.4)\\
&= \frac{p(O_A | O_B, l(A) = l(B))p(O_B)}{p(O_A)p(O_B)} && (3.5)\\
&= \frac{p(O_A | O_B, l(A) = l(B))}{p(O_A)}, && (3.6)\\
& && (3.7)
\end{aligned}
$$



(a) (b) (c) (d)

Figure 3-3: *This figure shows images from two cameras viewing the same area. (a) shows the reference image with a point marked. (b) shows the second image. (c) and (d) are the second image with a distribution superimposed of the possible locations of the corresponding observation given epipolar constraints and homography constraints respectively.*

where O_A refers to the observation in camera A and $l(A)$ is the unique label of observation O_A. This assumes that the locations of two observations of different objects in different sensors are independent. Thresholding this value allows one to make an instantaneous judgment of whether two observations resulted from the same object in the world. $p(O_A)$ can be computed simply by measuring the likelihood of observing an object at any particular location. $p(O_A|O_B)$ is the correspondence model. In many cases, $p(O_A|O_B)$ for a given value of O_B would be well-approximated by a unimodal Gaussian or a multi-modal Gaussian because objects tend to lie near surfaces.

This function could be estimated non-parametrically. Unfortunately, there will be false correspondence pairs and we do not want to rely on seeing a sufficient amount of correspondences in the reference camera to reliably estimate a two dimensional probability distribution on the second camera conditioned on each point in the first camera. Hence, parametric approaches which take advantage of the geometry of the real world are advantageous.

The first subsection discusses full external camera calibration using objects as correspondence points. This can only be accomplished when the location of the centroid of objects can be determined reliably in two cameras simultaneously *and* those detections include objects spanning a three dimensional subspace relative to the noise in measurements. We briefly discuss this case and show a simple example, but as this is not a focus of this thesis and is a relatively rare case, we do not cover details of the computation.

The second subsection discusses the common case where two cameras overlap but there are not enough reliable correspondences to reliably estimate external camera calibration. In this case, homography correspondences have been proven useful. A

homography is a one-to-one mapping of points in one camera to points in the other camera assuming that the points lie on a plane and each camera uses perspective projection.

The third subsection is concerned with the case of a pair of cameras with no overlap. This case is generally the most common.

Observation correspondence pairs

Calibrating cameras using correspondence of visual features is an extremely difficult problem in computer vision. Most of this difficulty arises from the correspondence problem (e.g., knowing which feature corresponds to which feature in a set of images). Once this correspondence can be established, there are established methods for calibrating cameras.

Without knowing something about the relative camera positions, this task is nearly impossible because searching the space of all possible correspondence mappings is computationally infeasible. Many variants of RANdom SAmpling and Consensus (RANSAC) algorithms [13] have been created to attempt to solve this problem reliably.

Thankfully, tracking correspondences are very different in character. First, the number of tracking correspondence pairs increases with the amount of time that tracking occurs. If there is not enough data to calibrate properly, one needs only wait until additional objects pass through that area of the environment. This means that the density of correspondence points can be much greater than the density of pixels.

Second, potential correspondences must occur at the same time. One need not consider potential correspondences of observations that occur at different times. If multiple objects are being tracked at once, one must search over the possible correspondence matchings of those pairs. Of course, as per the previous point, this data could be neglected in favor of data captured when only a single object was being tracked in both cameras.

Third, instantaneous and continuous correspondence are coupled. Correspondences of tracked objects that do not match for the entire tracking sequence need not be considered. Tracking can also be improved by considering the observations of the objects in multiple cameras if an effective model of correspondence is available. The following sections describe how to derive these models starting with the most restrictive and most powerful case.

3.2.1 External camera calibration

In the ideal case, the available object correspondence pairs enable external camera calibration. Given the internal parameters, the relative locations and rotations of the camera can be estimated. Faugeras [12] and Hartley [18] did early work on camera calibration for computer vision. Bundle adjustment is a well-established technique for determining the camera geometry given the correspondence pairs (see [58] for a survey).

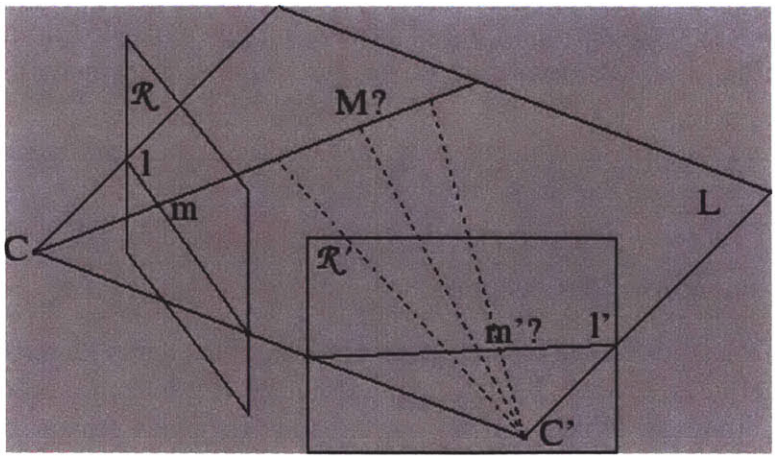


Figure 3-4: *This figure shows the epipolar constraints on two calibrated cameras.*

If external calibration is possible and correspondence can be reliably estimated, the positions of the cameras and the positions of the objects in the region of redundancy can be determined up to a single Euclidean transformation. Apart from a translation, rotation, and scaling, the exact position of the objects and cameras in the world would be known. Given the estimates of these parameters, normalized measurements of object height, approximate area, speed, and direction could be extracted. These normalized measurements could be used to build general models of appearance, size, and behavior that would transfer from one environment to another.

Unfortunately, external calibration is difficult from noisy estimates of object centroids. Our experimentation in this area has not proven robust to noise in our measurements of object centroids, regular errors caused by shadows or occlusions, and degenerate data which lies mostly on a single plane.

Given a calibrated camera system, what is known about the two cameras is the epipolar geometry. Given the location of an observation in one camera, m, the corresponding point in the second camera, m', is restricted to lie on (or near) the corresponding epipolar line given by the epipolar constraint

$$mFm' = 0 \tag{3.8}$$

where F is the fundamental matrix. This is illustrated in Figure 3-4. As was stated earlier, the locations of objects in the world tend not to occupy the majority of the three dimensional space. Even in the case of a fish tank or birds flying, it is not possible for fish to move outside of the tank or birds to fly through buildings or solid ground. Further, most cases are more restrictive because most active objects are constrained by gravity.

Because this is often the case, we often fit the reconstructed data with planes in space. This results in a homography of points in one camera to points in the other camera.

$$m' = Hm \tag{3.9}$$

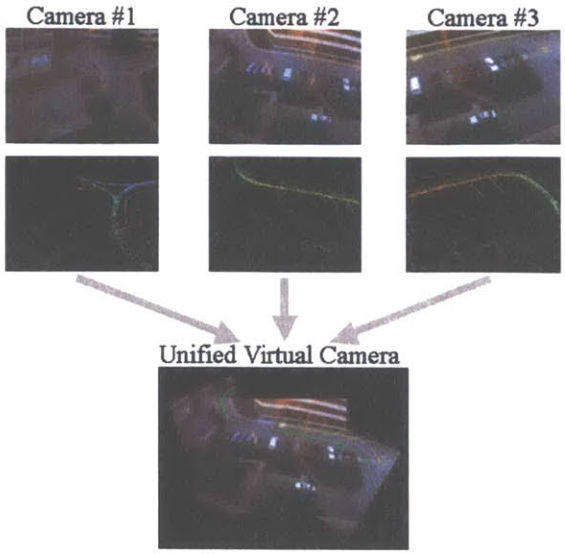


Figure 3-5: This figure illustrates how a correspondence model of three overlapping cameras allows an object to be tracked through multiple sensors in an extended environment.

3.2.2 Planar correspondence models

Because external calibration is not always possible and often not robust, we have investigated estimating a less complex model of correspondence directly– the homography. A homography is a projective mapping of points in one camera to points in the other camera. Usually homographies are used to represent correspondence between two views of points that lie on a plane.

In previous work[35], we have shown that it is possible to robustly estimate a single homography that maps tracked objects from one scene to another assuming those objects lie on a *single* ground plane in such cases where a single plane exists in the area of overlap between two cameras. This allows us to estimate correspondence between the tracking sequences in multiple cameras.

This section discusses extending this simple method to more general situations where multiple planes of correspondence are visible in more than one camera. These planes can result from the existence of multiple true planes of correspondence from multiple surfaces (e.g., a shopping mall) or objects of vastly different heights. Also, different sources of error in the estimates of the centroids (e.g., shadows, occlusions, adverse tracking conditions, etc.) can cause regular correspondence errors.

This estimation would necessitate simultaneously estimating outlier points, a segmentation, and the underlying homographies. This more general solution would enable establishment of correspondence anywhere piece-wise linear projections are valid. Further, the segmentation and homographies may be useful for directly inferring some functional understanding of the scene.

Robust estimation of a single plane of correspondence

In previous work [35, 56], we have illustrated the ability to calibrate cameras that are very far apart using object centroid estimates as correspondence points. Figure 3-5 shows a three camera views, the corresponding data, and the resulting unified reference frame with a single object being tracked through the entire scene.

Two corresponding points, m and m' are related by a homography transformation as follows.

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \propto \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.10}$$

Appendix B shows the details of estimating a homography from correspondence points.

It is potentially possible to improve the sampling of points that are used to hypothesize homographies in a number of ways. First, the second tracking sequence can be interpolated to better estimate correspondences to the exact times of the reference sequence frames. Second, points can be sampled more locally in space. Arguably this is not an advantage when the entire scene is modeled by single homography. But in the applications which potentially contain multiple planes, this would effectively reduce the number of false correspondences. It is also possible to use points from sequences that are at least locally consistent. Finally, it is possible to use point pairs from sequences which exhibit at least one very similar correspondence (and hence may be of similar height) increase the chances of determining a homography that matches the data.

Robust estimation of multiple planes of correspondence

Unfortunately, in most tracking environments, a single plane is often not sufficient to model the correspondences expected in the scene. This may be the case if the points are actually on multiple different planes or if errors in centroid estimates that are roughly linearly related to the position of the object are present. For example, if a person is occluded by a in one camera, the centroid of the person will be linearly biased by the amount of occlusion the person is undergoing, but the second camera will not be affected. This is a case where a homography can be useful even though there is no true analogous point in three dimensional space that corresponds to both centroids.

3.2.3 Cameras with no visual redundancy

The final case is both the most common and most difficult to detect. This is the case where two cameras have no visual overlap. For instance, a camera tracking in Rome, Italy and another in Boston, Massachusetts will never see the same object at the same time. Unfortunately, there will be a parametric model of correspondence

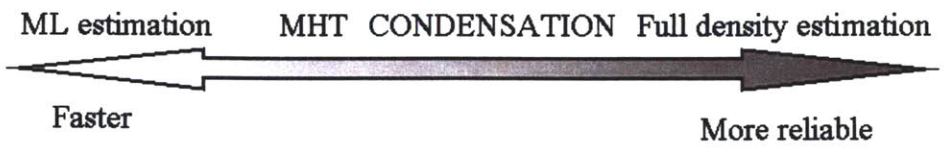


Figure 3-6: *This figure the speed and effectiveness of four types of state estimation including, maximum likelihood estimation, multiple hypothesis tracking, CONDENSATION, and full density estimation.*

that may be predictive of locations that co-occur. We do not discuss how this would be accomplished, but recognize the value of this detection process in automatically modeling multiple camera correspondence.

3.2.4 Future work

Experimentation in a wide range of environments is required to determine how effectively cameras with no visual redundancy can be characterized. With multiple planes of correspondence, one could determine which planes are parallel and determine where planes intersect. This information could be useful in modeling a scene, understanding more about the activities in a scene, and "passing off" from one correspondence model to another.

3.3 Continuous correspondence

Perhaps the most important type of correspondence for determining sets of observations that correspond to the same object in the environment is continuous correspondence. Continuous correspondence involves estimating the state of an object continuously through time given noisy observations. Correspondence techniques employ various levels dynamics modeling in this regard.

Also, there is a gradation of discreteness of state estimation in tracking continuously through time. On the one extreme is maximum likelihood state estimation where only the most likely state of the object in each frame is estimated. On the other extreme is a full density estimation. Figure 3-6 illustrates that there is a tradeoff that is made in reliability and speed.

Maximum likelihood estimation is the fastest technique. It involves simply estimating the most likely state at each point (with a local search) in time given the previous estimate and the current observation. Multiple hypothesis tracking adds the complexity of considering multiple possible states when uncertainty exists. CONDENSATION takes this a step further by estimating a sampled density over the possible states. Full density estimation is simply evaluating the full posterior of the state at every time step.

We have chosen a simple implementation of multiple hypothesis tracking (MHT). Below we explain our method, implementing a version of MHT is not a focus of this thesis.

3.3.1 Multiple Hypothesis Tracking

Establishing correspondence of object observations between frames is accomplished using a linearly predictive multiple hypotheses tracking algorithm which models both object position and object size. We have implemented an online method for seeding and maintaining sets of Kalman filters.

At each frame, we have an available pool of Kalman models corresponding to each object being tracked and a new available pool of object observations that could be observations of those objects in the current frame. First, the models are probabilistically matched to the object observations that they could explain. Second, the connected regions which could not be sufficiently explained are checked to determine if new Kalman models should be initialized. Finally, models whose fitness (as determined by the inverse of the variance of its prediction error) falls below a threshold are removed.

Matching the models to the object observations involves checking each existing model against the available pool of object observations which are larger than a pixel or two. All matches with relatively small error are used to update the corresponding model. If the updated models have sufficient fitness, they will be used in the following frame. If no match is found a "null" match can be hypothesized which propagates the model as per its dynamics and decreases its fitness by a constant factor. If the object reappears in a predictable region of uncertainty shortly after being lost, the model will be re-established. Because our classification system requires tracking sequences which consist of representations of a single object, our system generally breaks tracks when objects interact rather than guessing at the true correspondence when it is not certain.

The unmatched models from the current frame and the previous two frames are then used to hypothesize new models. Using pairs of unmatched object observations from the previous two frames, a model is hypothesized. If the current frame contains a match with sufficient fitness, the updated model is added to the existing models. To avoid possible combinatorial explosions resulting from noise in the estimation of object observations, it may be desirable to limit the maximum number of existing models by removing the least probable models when excessive models exist. In noisy situations (e.g. ccd cameras in low-light conditions), it is often useful to remove the short tracks that may result from random correspondences. Further details of this method can be found at `http://www.ai.mit.edu/projects/vsam/`.

3.4 Discontinuous object correspondence

Once the continuity of observations has been broken, it is necessary to rely on longer-term characteristics of the object. This is not a focus of this research, but it will be discussed in the following three sections.

3.4.1 Dynamics-based models

The first type of discontinuous correspondence modeling is a short-term estimate of the object's state given the current object dynamics. In some environment, short occlusions or distractions can result in an object being tracked as two separate objects. The multiple hypothesis tracker mentioned in the previous section already exploits this type of regularity.

3.4.2 Appearance-based models

The second type of discontinuous correspondence modeling involves estimating the appearance of the object. This can be accomplished by a correlation model, color histogram model, or any other type of appearance-based model. The following chapter introduces an method for decomposing the description of pedestrian images into component regions that exhibit regularity. The resulting concise description of the pedestrian could be useful in this regard.

3.4.3 Behavioral models

The final type of discontinuous correspondence modeling involves characterizing the activities that objects may be performing. If by concatenating two tracking sequences, the resulting tracking sequence is consistent and not detected as an atypical observation in an environment, it is likely that the two sequences resulted from the same object in the world. The following two chapters will discuss way that objects appearance, shape, and activities can be characterized as well as how anomalous activity sequences could be determined.

3.5 Normalizing regular properties of objects

Under certain circumstances, some properties of objects can be normalized. For instance, the height of an object can be normalized by assuming that they objects lie on a plane in the world and are being recorded through a perspective transformation. While this is not a focus of this thesis, it is an important consideration in modeling in multiple environments and building concise models of object appearance and activities.

3.6 Tracking discussion

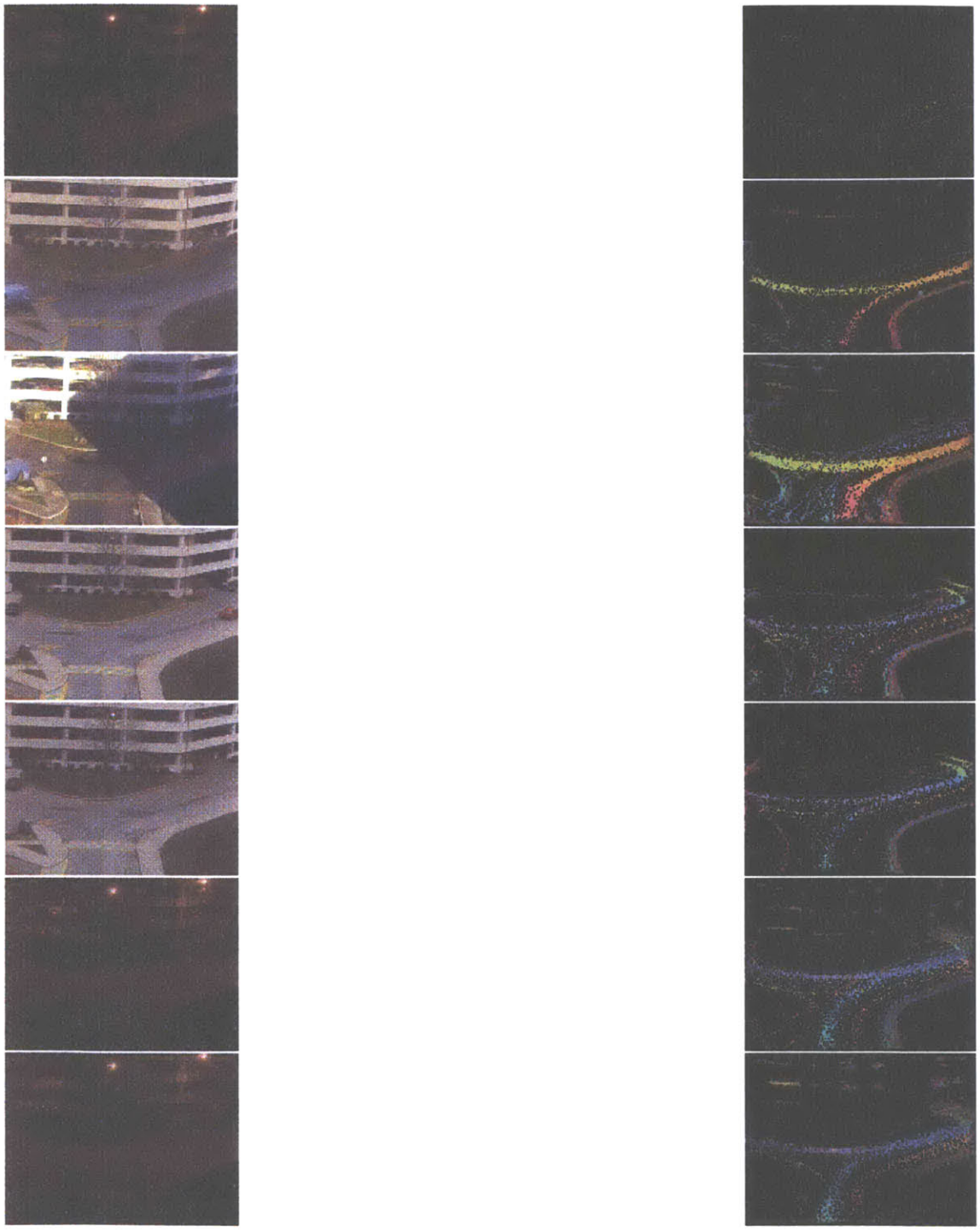
On an SGI O2 with a R10000 processor, this method can process 11 to 13 160x120 frames a second. On a 1Ghz Pentium 3 processor, the method can process 20-25 320x240 frames a second. The variation in the frame rate is due to variation in the amount of foreground present. Our tracking system has been effectively storing tracking information for five scenes since 1997[61]. Figure 3-7 and figure 3-8 show accumulated tracks in two scenes over the period of a day. While quick changes in

cloud cover (relative to α, the learning rate) can sometimes necessitate a new set of background distributions, it will stabilize within 10-20 seconds and tracking will continue unhindered.

The tracking system has the most difficulty with scenes containing high occurrences of objects that visually overlap. The multiple hypothesis tracker is not extremely sophisticated about reliably disambiguating objects which cross. Adding more complex dynamics or appearance templates[16] could help in this regard. This problem can be compounded by long shadows, but for our applications it was much more desirable to track an object and its shadow and avoid cropping or missing dark objects than it was to attempt to remove shadows. In our experience, on bright days when the shadows are the most significant, both shadowed regions and shady sides of dark objects are black (not dark green, not dark red, etc.).

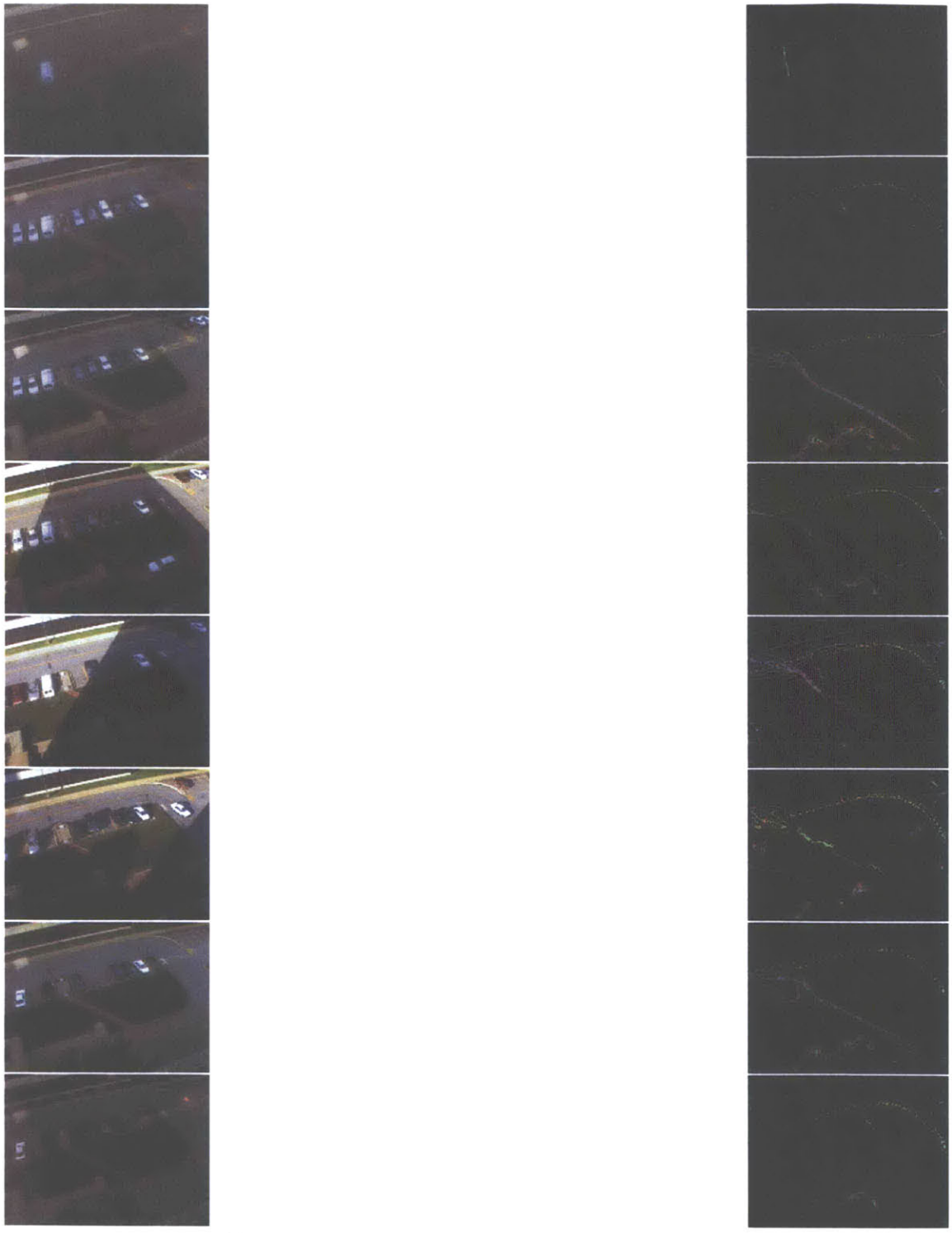
The tracker was robust to all but relatively fast lighting changes (e.g. flood lights turning on and partly cloudy, windy days). It successfully tracked outdoor scenes in rain, snow, sleet, hail, overcast, and sunny days. It has also been used to track birds at a feeder, mice at night using Sony NightShot, fish in a tank, people in a lab environment, and objects in outdoor scenes. In these environments, it reduces the impact of repetitive motions from swaying branches, rippling water, specularities, slow moving objects, and acquisition noise. The system has proven robust to day/night cycles and long-term scene changes.

This chapter and the last explain how we can reliably determine sets of observations of the same object in the environment (Multiple Observation Sets) over long periods of time, across multiple cameras, and in widely varying conditions. The following chapter discusses how MOSs can be exploited to estimate models of classes of object observations. Without MOSs we would be required to rely on a standard unsupervised clustering mechanism (e.g., a mixture of Gaussians). We will show an approach that relies primarily on co-occurrence of object observations in MOSs to derive rich models of object type and object activities.



(a) (b)

Figure 3-7: *This figure shows consecutive hours of tracking from 6am to 9am and 3pm to 7pm. (a) shows the image at the time the template was stored and (b) show the accumulated tracks of the objects over that time. Color encodes object direction and intensity encodes object size. The consistency of the colors within particular regions reflects the consistency of the speed, direction, and size parameters which have been acquired.*



(a) (b)

Figure 3-8: *This figure shows consecutive intervals of tracking on a different scene than previous figure. Also, this particular day was foggy, then clear, then overcast. As the templates show, the tracking was relatively unaffected.*

Chapter 4

Co-occurrence based clustering

This chapter discusses a method for clustering observations given sets of observations of the same object or class of object, referred to as multiple observations sets (MOSs). There are many situations where multiple observations of the same object or process are available rather than single observations. Observations may be grouped by hand or by any process which is allowed to make more than a single observation of a process. For instance, if an infant is given a new object, it makes many observations of the object through manipulation using many modalities (sight, touch, smell, taste). Except in the case of certain psychological experiments, almost all objects a human being observe in the world can provide sets of observations.

Chapter 3 discussed a method for passively determining sets of observations that correspond to the same moving object. Our multiple camera adaptive background tracking system described in the previous two chapters has tracked over 10 million objects since 1997. The resulting 10 million MOSs are much more informative than the corresponding billion or so individual observations. Each MOS exhibits some of the variation that one type of object undergoes. While some clustering mechanisms rely on separable densities or a continuous, separable manifold of examples, our method relies on the variability in the MOSs.

Consider the millions of silhouettes of tracked objects in an urban environment. Given a set of 400 prototype silhouettes that are representative of the millions, some prototype silhouettes would be similar to pedestrians of different sizes and in different positions. Others would be similar to different vehicles of different sizes at different angles. Over time, tracked objects will tend to present *EITHER* like the set of pedestrian prototypes *OR* like the set of vehicle prototypes but rarely both. For example, it is certainly possible to see two views of a car at different angles under different lighting in a single MOS, but unless our tracking system has made a grave error, there should never be an observation of a car and a person in the same MOS. This information is what enables Multiple Observation Learning.

This chapter discusses how Multiple Observation Learning (MOL) exploits the additional information available in MOSs. Our method uses Vector Quantization (VQ) to develop a codebook of observations that are representative of the entire set of observations. Using this codebook to represent our continuous observations, we can accumulate joint co-occurrence statistics of the observations in the codebook that

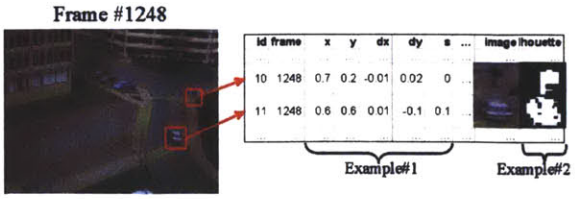


Figure 4-1: *This figure shows a single frame from a typical scene and the information which recorded for the two moving objects. The fields which are used for the two classification examples are labeled.*

tend to occur within the same MOSs. Finally, we perform hierarchical clustering of the observations in the codebook using the accumulated co-occurrence data. This classification probabilistically clusters the codebook observations into sets of observations that tend to occur in the same MOSs and separates codebook observations that do not generally occur in the same MOSs. In our silhouette example described above, it would probabilistically cluster codebook observations into pedestrian and vehicle clusters.

This chapter begins with some discussion on previous work in clustering. In Section 4.1, Multiple Observation Learning (MOL) is introduced using an artificial example with discrete integer observation values from one to ten. This section covers accumulating co-occurrence statistics, estimating the model parameters, and classifying examples given the model. Next, a continuous example is introduced to explain the complexities of using this algorithm with continuous-valued observations. This section explores the trade-offs of different methods of discretizing continuous observations. Also, issues and assumptions are covered including computational complexity, storage complexity, convergence, the assumption of independence of samples in MOSs, normalization techniques, and identifiability.

Section 4.5 shows some examples of classification on tracking data. As shown in Figure 4-1, for every frame that an object is detected its unique identifier (assigned in the last chapter), its location (x,y), speed/direction (dx,dy), and size are recorded. Also, an image of the object and a binary motion silhouette are cropped from the original image and the binary difference image respectively. As mentioned in the previous chapter, some of these values can be normalized and mapped to a more global (universal) coordinate system. Two sets of experiments are discussed that perform classification based on the {x,y,dx,dy,size} representation and the binary motion silhouette representation using literally millions of training examples. These experiments result in a concise description of the clusters of activities and object types respectively.

The remainder of this chapter discusses a co-occurrence based image representation and its application to detection, alignment, and factorization of a class of images (pedestrian images). The factorization mechanism is nearly identical to the previous co-occurrence based factorization. Appendix C has details of the computation in the form of pseudo-code for the benefit of the reader.

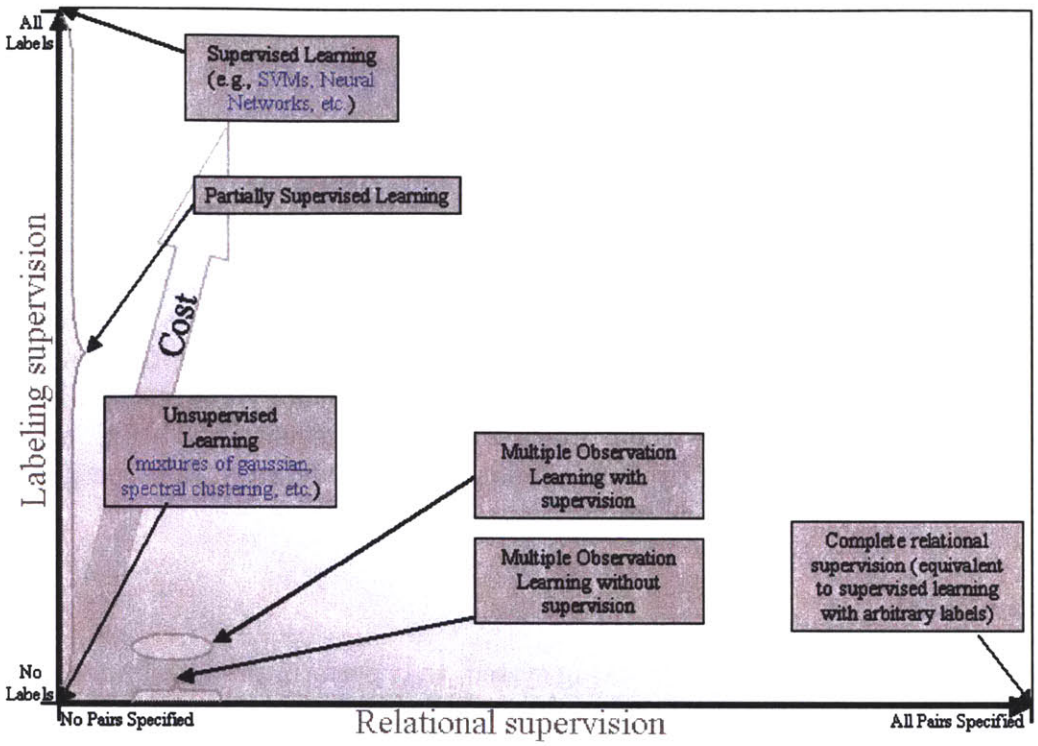


Figure 4-2: This figure shows relational supervision vs. labeling supervision.

4.1 Background in clustering and classification

The different types of clustering and classification can be characterized by the type and amount of supervision available. Given only a set of data points and no supervision, one is restricted to unsupervised learning approaches. Unsupervised classification can be effective in clustering data according to their class if the data fits a simple parametric model or is relatively separable. Unfortunately, in our experience with perceptual data this is rarely a reasonable assumption.

Supervision for classification tasks generally refers to supplying class labels for the data points. Given class labels for the set of data points, supervised classification attempts to determine a function that indicates the class of any new data point. *Relational supervision* refers to supplying relational information about pairs of data points. In our case, a correspondence system establishes identity relationships between the data points.

Figure 4-2 shows many different classification approaches plotted based on the amount of relational and labeling supervision they employ. The vertical axis describes the amount of labeling supervision that is available. This covers a spectrum of approaches from completely unsupervised approaches where no labels are associated with any data points to completely supervised approaches where every data point is assigned the proper label. The horizontal axis describes the amount of relational supervision that is available. If all pairwise identity relationships are defined,

data points can be trivially grouped based on identity and this problem reduces to supervised learning with unknown labels.

There is an associated cost in creating the classification systems in figure 4-2.

$$C_{total} = C_{aquire} * numData + C_{label} * numLabels + C_{relation} * numRelations \quad (4.1)$$

where C_{aquire} is the cost of acquiring data, C_{label} is the cost of labeling supervision, $C_{relation}$ is the cost of providing relational supervision. Our goal is to minimize this cost while maximizing the functionality of our system. In the case of perceptual data, the cost of acquiring data and the cost of some relational labeling is extremely low while the cost of supervision is comparatively large. One can acquire billions of data points without intervention. One can obtain correspondence relationships between a significant portion of them without intervention. Unfortunately, labeling even one percent of those observations by hand would take a moderately talented graduate student months of continuous effort.

Previous work

There are countless examples of tracking system that perform predetermined classification tasks on tracked data, e.g. human vs. vehicle; walking vs. running[3]; walking, marching, line-walking, and kicking[10]; etc. Our system does not perform predetermined classification tasks.

Many generally applicable unsupervised clustering mechanisms assume a locally smooth manifold of points or that the entire densities are separable. In practice, we have learned that these are particularly poor assumptions for object observations (e.g., images, silhouettes, positions, velocities, sizes, or the entire description) because of poor lighting, occlusions, and other factors in acquiring real-world data. In our outdoor, 24/7 tracking system, all object types will have some observations that can only be described as small, dark, noisy, erratic blobs.

Our method is similar to the work of Johnson and Hogg [27]. They begin their process by on-line Vector Quantization on the input space. They then quantize again into a predetermined number of probability mass functions (pmfs) over their discrete states. While a significant number of these pmfs will result in tight clusters of activity and shape, it is unclear how to relate two inputs that are grouped into separate pmfs or to select the proper number of pmfs.

Our hierarchical classification involves a step that has the flavor of Normalized Cuts and its many derivatives (see [51]). It has discrete nodes (defined by the codebook prototypes). It has edges which represent pair-wise distances (or dissimilarities or costs) between them. In addition, the goal is to determine two sets of nodes that have the largest distances (or weighted distances) between them. However, that is the extent of the similarity. Our pairwise measurements are probabilities, not "distances." Those similarities are not directly related to the coordinates or properties of the nodes, but rather are a function of the co-occurrence of the observations. Our "cut" does not produce two discrete sets that minimize the cut "similarities." It produces two distributions that both explain the observed joint statistics and are

relatively dissimilar.

Our method can be described as estimating a joint distribution as a mixture of products of marginal distributions. Recently, Lee and Seung citekey:seung have popularized this approach. Thomas Hofmann's previous probabilistic treatment is very closely related to our approach. The statistics community has used similar approaches in the past [11]. Neither of these previous approaches dealt with the complexity of continuous observations.

4.2 MOL in discrete observation spaces

We will describe Multiple Observation Learning in discrete spaces using a simple, artificial example. This example involves sets of integer observations from one to ten. The next section introduces a continuous observation example and discusses the complexities involved in effectively modeling co-occurrences with continuous observations.

4.2.1 A simple, discrete example

Given the experiment:

- Place N individuals in a room. Ask individuals to approach the tester and name S numbers from 1 to 10 and then sit down again. Repeat this test T times.

This experiment would result in T MOSs on the space of [1,10], e.g., {1,4,3,9,4,2}. This is similar in character to the perceptual observations produced by our attention and correspondence systems. The goal is to estimate the number of people, their preferences, and how likely they are to participate in the experiment from the pieces of paper given.

If the MOS size, S, was one, this would be an unsupervised data set. Each MOS would contain a single observation that could have been sampled from any individual. Nothing could be determined about the individuals in the room except their cumulative probability of choosing any number from one to ten.

If S was very large and the test subjects had individual preferences on the numbers that they produced that could be represented by a probability mass function (pmf), the pmf's of each MOS of a particular individual would closely approximate that person's pmf. If the individual's preferences were sufficiently different from each other relative to the variation resulting from the sampling process, the pmfs could be effectively clustered. Hence, it would be possible to determine both the number of individuals (number of clusters), their likelihood of participating in the experiment (number of samples in each cluster), and their number preferences (the average pmf of the cluster). This could even be done if there were many individuals participating in the test.

Unfortunately, for most problems S must be very large to create a characteristic pmf that can be effectively clustered and often one cannot control the value of S. As S decreases, clustering becomes increasingly unreliable. For instance, in the extreme

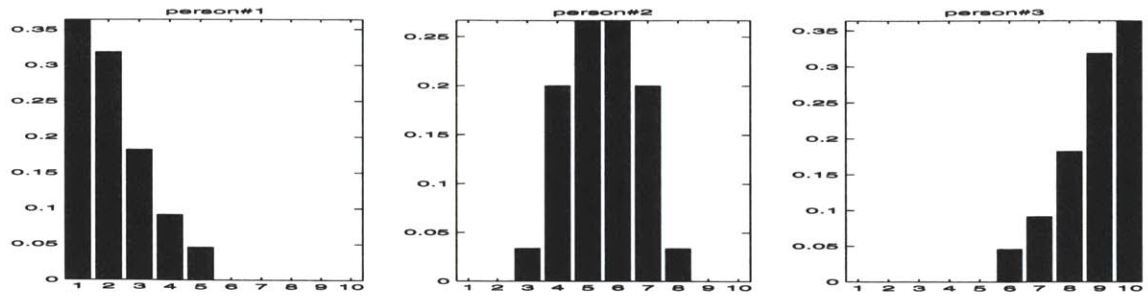


Figure 4-3: This figure shows three pmf's for three individual's preferences on numbers. The first person prefers low numbers, the second person prefers middle numbers, and the third person prefers high numbers.

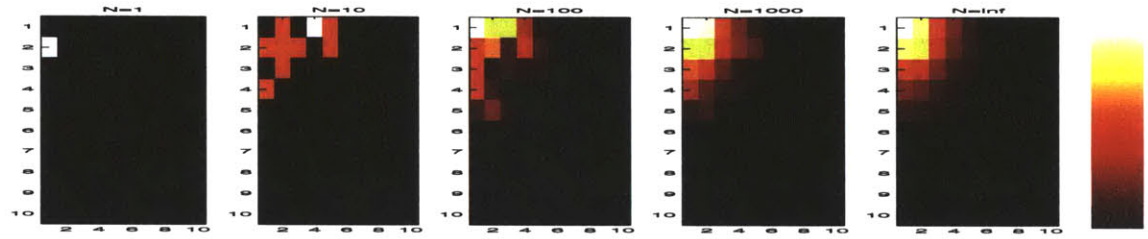


Figure 4-4: This figure shows C after a single sample, ten samples, 100 samples, and 1000 samples as well as the theoretical limit as the number of samples approaches infinity. Brighter values are higher.

example, you may get only a pair of samples from each test subject. But under a stronger set of assumptions than the previous case, it is still possible to determine the number of individuals, their likelihood of participating in the experiment, and their individual preferences of producing the numbers from one to ten.

To illustrate this case, we will assume the sample size is two ($S = 2$) and the number of sample sets T is infinite. We will look at two cases involving the three individuals with profiles shown in Figure 4-3. These three people prefer low, medium, and high values respectively.

Consider the case where the first person is the only individual in the room. We assume each observation is an independent and identically distributed (IID) sample from his pmf. We measure the co-occurrence of each pair of numbers in a 10x10 matrix. By normalizing this matrix by the number of observation pairs(T), this resulting matrix C is a joint probability density function of the observation pairs, x_0 and x_1, e.g.,

$$C_{i,j} \equiv p(i,j) = \frac{T(i,j)}{T} \tag{4.2}$$

where T is the number of pairs (MOSs) and $T(i,j)$ is the number of i,j-pairs observed in the samples. Figure 4-4 shows C after a single sample, ten samples, 100 samples, and 1000 samples as well as the theoretical limit as the number of samples approaches

infinity. In the limiting case with only the first person participating, elements of the ideal co-occurrence matrix are

$$
\begin{aligned}
C^*_{i,j} &= \lim_{T\to\infty}\left(\frac{T(i,j)}{T}\right) && (4.3)\\
&= \lim_{T\to\infty}\left(\frac{\frac{T_1 T_1(i,j)}{T_1}}{T}\right) && (4.4)\\
&= \lim_{T\to\infty}\left(\frac{T_1}{T}\frac{T_1(i,j)}{T_1}\right) && (4.5)\\
&= p(c_1)p(i,j|c_1) && (4.6)\\
&= p(c_1)p(i|c_1)p(j|c_1) && (4.7)\\
&= p(i|c_1)p(j|c_1) && (4.8)
\end{aligned}
$$

where T_1 is the number of observation pairs from person one, $T_1(i,j)$ is the number of i,j-pairs sampled from person one, $p(c_1)$ is the probability of person one being the individual that provided an observation pair (in this case 1), $p(i,j|c_1)$ is the joint probability of person one providing an i,j-pair, and $p(i|c_1)$ is the probability of person one drawing sample i. [1]

Now, consider the case were all three individuals are in the room. We can accumulate the co-occurrence statistics of the pairs provided by all three individuals. The co-occurrence matrix would be

$$
\begin{aligned}
C^*_{i,j} &= \lim_{T\to\infty}\left(\frac{T(i,j)}{T}\right) && (4.9)\\
&= \lim_{T\to\infty}\left(\frac{T_1(i,j)+T_2(i,j)+T_3(i,j)}{T}\right) && (4.10)\\
&= \lim_{T\to\infty}\left(\sum_c \frac{\frac{T_c T_c(i,j)}{T_c}}{T}\right) && (4.11)\\
&= \lim_{T\to\infty}\left(\sum_c \frac{T_c}{T}\frac{T_c(i,j)}{T_c}\right) && (4.12)\\
&= \sum_{c=c_1}^{c_3} p(c)p(i,j|c) && (4.13)\\
&= \sum_{c=c_1}^{c_3} p(c)p(i|c)p(j|c). && (4.14)
\end{aligned}
$$

where c indexes the person. This is simply the sum of the three joint distributions that would result from single-person experiments weighted by their probability of participating. Figure 4-5 shows limit of the co-occurrence matrix if all three individuals

[1] C is simply the outer product of the pmf of person one with itself. i.e., $C = p(i|c)^T * p(i|c)$.

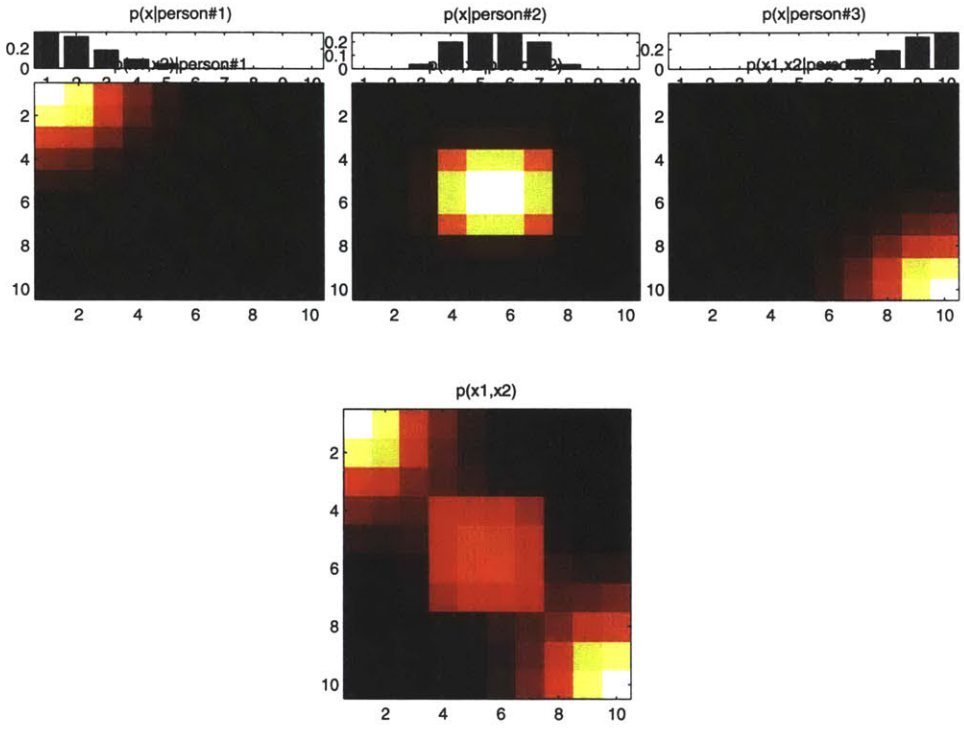


Figure 4-5: The co-occurrence matrix which would result from the three single person experiments as well as the experiment with all three individuals. Brighter values are higher.

participate equally in the experiment as the number of samples approaches infinity ($T \rightarrow \infty$).

The goal of MOL is to estimate parameters which are consistent with the co-occurrence statistics. Given the number of people, it is possible to iteratively estimate the $p(c)$ and $p(i|c)$ for each person c that minimize the error between the measured co-occurrence and the estimated co-occurrence. By analyzing the error for models containing different numbers of people, it is possible to estimate the correct number of people. Obviously, this is not always possible. For instance, if two people have the exact same preferences, they will be indistinguishable. This and other issues will be investigated further in Section 4.2.5.

The following three subsections cover MOL in discrete spaces. Subsection 4.2.2 discusses how to accumulate co-occurrence statistics given a set of MOSs. Subsection 4.2.3 discusses estimation of the latent class parameters as well as determining hierarchical sets of class models. Subsection 4.2.4 discusses how to use the resulting latent class models to classify MOSs. Subsection 4.2.5 discusses other considerations in discrete MOL.

4.2.2 Accumulating co-occurrence statistics

Our model for the production of an MOS is simple. There are N underlying latent classes, each of which is sampled with some prior probability, $p(c)$. An object of class c, when observed in a camera, produces an observation given some probability distribution, $p(i|c)$. As long as the object is observed, it will produce independent observations from the same distribution. This model reflects our assumption of the independence of samples in an MOS.

Our method disregards temporal information in the MOSs and considers them as multi-sets of independent observations. A multi-set is a set which can contain multiple instances of the same element (e.g., $(1, 2, 4, 2, 8, 7, ...)$). Each (ordered) pair within an MOS (excluding pairing observations with themselves) is evidence that those observations may result from the same underlying class.

The m multi-sets of observations $\{M^1, M^2, ...M^m\}$ are used to estimate a co-occurrence matrix, C where $C_{i,j}$ is the estimated probability that a pair of observations $\{O_i, O_j\}$ will be from an MOS chosen at random from the m multi-sets.

We have covered the case where all MOSs are pairs of observations ($S = 2$) and the probability of drawing from each MOS is uniform. If the size of the multiple observation sets, S, varies for each sample or some MOSs are more likely to be drawn, accumulating the co-occurrence statistics must be computed as follows.

First, the matrix of the co-occurrences, C, is initialized to zeros or a prior joint distribution. Given a multi-set, the element $C_{i,j}$ corresponding to each possible pair of observations (excluding pairing observations with themselves) is incremented by the inverse of the number of valid pairs in that MOS multiplied by the probability of drawing from that MOS. Given an MOS, $M^u = \{x_1, x_2, ..., x_T\}$, the element $C_{i,j}$

corresponding to each pair $\{x_a, x_b \; where \; a \neq b\}$, is incremented by w

$$w = \begin{cases} \frac{p(M^u)}{(|M^u|^2 - |M^u|)} & \text{if } a \neq b, x_a = o_i, x_b = o_j \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

where $p(M^u)$ is the probability of drawing from M^u and $(|M^u|^2 - |M^u|)^2$ is the number of valid pairs in this sequence. Thus

$$C_{i,j} = \sum_u \sum_{x_a \in M^u} \sum_{x_b \in M^u} w \tag{4.16}$$

By this definition, the co-occurrence matrix is a valid symmetric joint distribution over the observation values.

In most cases, $p(M^u)$ is assumed to be uniform $(\frac{1}{m})$. This weighs each MOS equally in the estimation of C. While this was done for our experiments, it would be possible to weigh the contribution of each MOS. Factors such as the number of observations, the number of sensors used to collect them, and the certainty of their correspondence could be used to reduce the effect of unreliable and unrepresentative MOSs in the estimation of C.

If there was a single underlying class c_0 and infinite MOSs to train, $C_{i,j}$ would converge to $p(i|c_0)p(j|c_0)$. With N underlying classes,

$$\lim_{T \to \infty} C_{i,j} = \sum_{c=1}^{N} p(c) * p(i|c) * p(j|c). \tag{4.17}$$

Online Estimation

For extremely large streams of data, a running estimate of the co-occurrence is useful. Without examining the entire dataset, it is not possible to determine exact values for $p(M^u)$. Online estimation involves aggregating the values into a matrix that is proportional to C. By using a value $\hat{p}(M^u)$ which is proportional to the correct value of $p(M^u)$, an accumulation matrix can be estimated that is proportional to C,

$$C_{i,j}^{accum} = kC_{i,j} = \sum_u \sum_{x_a \in M^u} \sum_{x_b \in M^u} kw \tag{4.18}$$

C is obtained by normalizing C^{accum} by the proportionality constant (k). For the uniform case, $\hat{p}(M^u) = 1 = k\frac{1}{m}$ (or any constant value) and hence the co-occurrence matrix is the accumulated matrix divided by the number of MOSs used to compute it. e.g.,

$$C = C^{accum}/m. \tag{4.19}$$

This can also be used to estimate the co-occurrence over an exponential window on the recent past. By increasing the proportionality constant by a factor of $(1 + \epsilon)$,

[2]Pairs are ordered so $\{a, b\}$ and $\{b, a\}$ are both valid pairings.

the current estimate of C is equivalent to the estimate which weighs observations inversely proportional to how long ago they occurred[3].

4.2.3 Estimating the latent classes

Our classification method attempts to estimate N latent class weights ($\hat{p}(c)$) and observation probability mass functions ($\hat{p}(i|c)$) which are consistent with the observed co-occurrence matrix, C. Given the latent class parameter estimates, the co-occurrence estimate is

$$\hat{C}_{i,j} = \sum_{c=1}^{N} \hat{p}(c)\hat{p}(i|c)\hat{p}(j|c). \tag{4.20}$$

Similar to Hofmann [19] and more recently Lee and Seung [33], we iteratively estimate the parameters that minimize the kl-divergence between C and $\hat{C}$

$$E = \sum_{i,j} \left(C_{i,j} \log \frac{C_{i,j}}{\hat{C}_{i,j}} \right). \tag{4.21}$$

The corresponding update rule for the weight of each class is

$$\begin{aligned}
\hat{p}'(c) &\propto \sum_i \sum_j \left(\frac{C_{i,j}}{\hat{C}_{i,j}} \hat{p}(c)\hat{p}(i|c)\hat{p}(j|c) \right) && (4.22)\\
&= \sum_i \sum_j \left(C_{i,j} \frac{\hat{p}(c)\hat{p}(i|c)\hat{p}(j|c)}{\sum_{c=1}^{N} \hat{p}(c) * \hat{p}(i|c) * \hat{p}(j|c)} \right) && (4.23)\\
&= \sum_i \sum_j \left(C_{i,j} \frac{p(c)p(i,j|c)}{\sum_{c=1}^{N} p(c)p(i,j|c)} \right) && (4.24)\\
&= \sum_i \sum_j C_{i,j} p(c|i,j). && (4.25)
\end{aligned}$$

This is the sum of each co-occurrence element, $C_{i,j}$ weighted by the probability of this particular latent class given that (i,j) observations pair. It is the proportion of the observed co-occurrences accounted for by the latent class c. The update rule for

[3]This requires occasional re-normalization of C^{accum} to avoid the proportionality constant increasing beyond the precision of the computer.

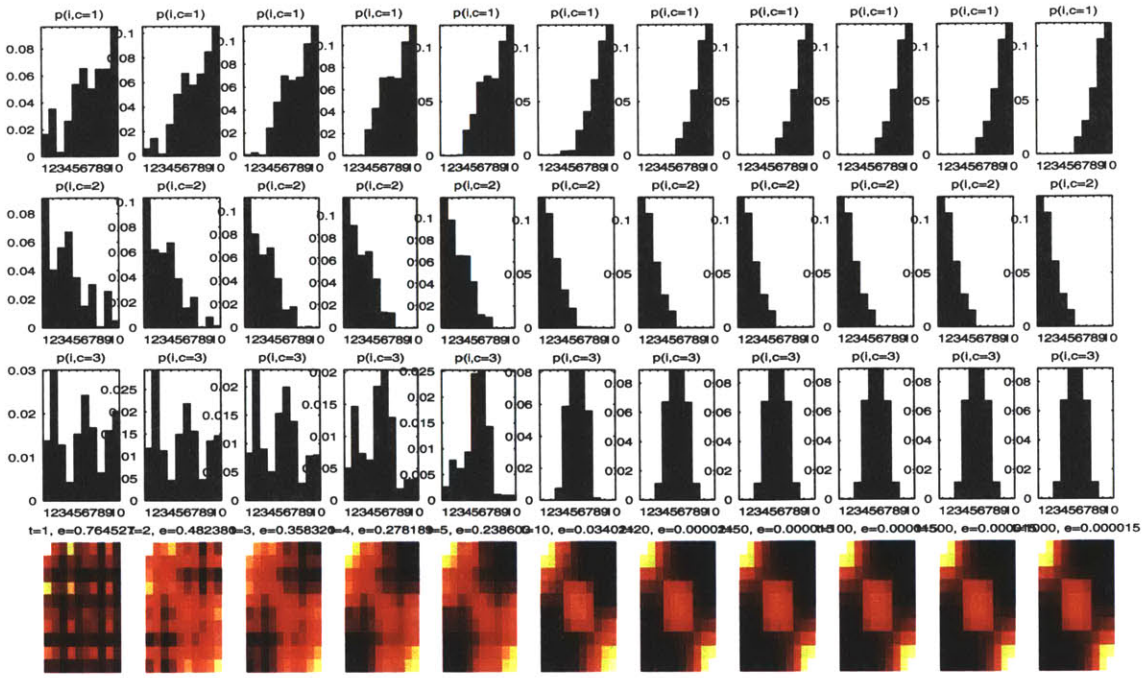


Figure 4-6: This figure shows the iterative estimation of the three latent class pmfs and the resulting estimated co-occurrence matrix after 1, 2, 3, 4, 5, 10, 20, 50, 100, 500, and 1000 iterations. After the first iteration the latent class pmfs are still somewhat random. By the fifth iteration, the latent class pmfs already resemble the generative classes used to estimate the co-occurrence matrix.

the class-conditional distribution is

$$\hat{p}'(i|c) \propto \hat{p}(i|c) \sum_j \left(\frac{C_{i,j}}{\hat{C}_{i,j}} \hat{p}(c)\hat{p}(j|c) \right) \tag{4.26}$$

$$= \sum_j \left(C_{i,j} \frac{\hat{p}(c)\hat{p}(i|c)\hat{p}(j|c)}{\sum_{c=1}^{N} \hat{p}(c) * \hat{p}(i|c) * \hat{p}(j|c)} \right) \tag{4.27}$$

$$= \sum_j C_{i,j} p(c|i,j) \tag{4.28}$$

which is the sum of the co-occurrence in row i weighted by the probability the latent class c given each co-occurrence pair in that row. For instance, if the latent class probability for class c for an entire row is approximately 1 (that class is most likely to explain all pairs in that row), $\hat{p}(i|c) \approx \sum_j C_i, j$, which is the measured marginal probability of observation i.

If the number of latent classes was known, this procedure could be used to estimate the parameters of the latent class. Figure 4-6 shows this iterative estimation for the three person problem introduced earlier. The co-occurrence matrix was estimated with 1000 pairs of observations. Starting from random initial conditions, it quickly converges to a global maximum. One thousand such trials all converged to the correct

set of latent class parameters within 100 iterations.

If the co-occurrence matrix is fit with two or four classes, one would get the results similar to those shown in Figure 4-7. For our simple experiment, the proper number of classes is the minimum number of classes that produce the minimal error. By associating a description cost to each latent class relative to the cost of estimation error, one can determine the number of classes that minimizes the description length of the co-occurrence data. This type of N-way clustering can be effective in situations where there are N independent classes of objects.

Hierarchical binary classification

An alternative to clustering into a pre-determined number of clusters is to use hierarchical clustering. This can be advantageous in situations where the number of classes is not known or when some classes are more closely related than others. For instance, people and vehicles are reasonably discrete classes while vehicles could be described by a single class or multiple similar classes (e.g, cars, vans, trucks, etc.). Therefore, one might expect that pedestrians and cars would represent discrete branches of a hierarchy while the different types of cars would be children of the vehicle class.

Given the entire co-occurrence matrix, two latent classes ($c_r \in \{0, 1\}$) are estimated that best approximate the observation co-occurrences as described above. These root latent classes are used to partition the co-occurrences into two co-occurrence matrices corresponding to the co-occurrence measurements attributed to each class.

$$C^0_{i,j} = C_{i,j} p(c_r = 0|i, j) \tag{4.29}$$

and

$$C^1_{i,j} = C_{i,j} p(c_r = 1|i, j) \tag{4.30}$$

These two co-occurrence matrices sum to the original co-occurrence matrix. The process is repeated on each new co-occurrence matrix producing two sets of new latent classes, $c_0 \in \{0, 1\}$ and $c_1 \in \{0, 1\}$. Each of these latent classes can be used to again partition the co-occurrences recursively.

$$\begin{aligned} C^{00}_{i,j} &= C^0_{i,j} p(c_0 = 0|i, j) && (4.31) \\ &= C_{i,j} p(c_0 = 0|i, j) p(c_r = 0|i, j) && (4.32) \end{aligned}$$

Figure 4-8 shows this procedure performed on our example problem. The first branch results in two latent class models: one for observations of person two and person three combined and one for person one. The weight of the first latent class is nearly twice the second. The pmf for the first latent class is approximately the sum of the pmfs from the two corresponding people. The branch on C^0 results in two latent classes that are approximately the same as the two corresponding individuals' pmfs. These three latent classes ($c_r = 1, c_0 = 0, c_1 = 1$) correspond to the three pmfs used to generate this data.

Once the parameters for the hierarchy of latent classes have been determined,

Figure 4-7: This figure shows the iterative estimation of the class-conditional pmfs and the resulting co-occurrence matrix assuming two classes (a) and four classes (b) after 1, 2, 3, 4, 5, 10, 20, 50, 100, 500, and 1000 iterations. (c) shows the log of the estimation error for N={2,3,4}.

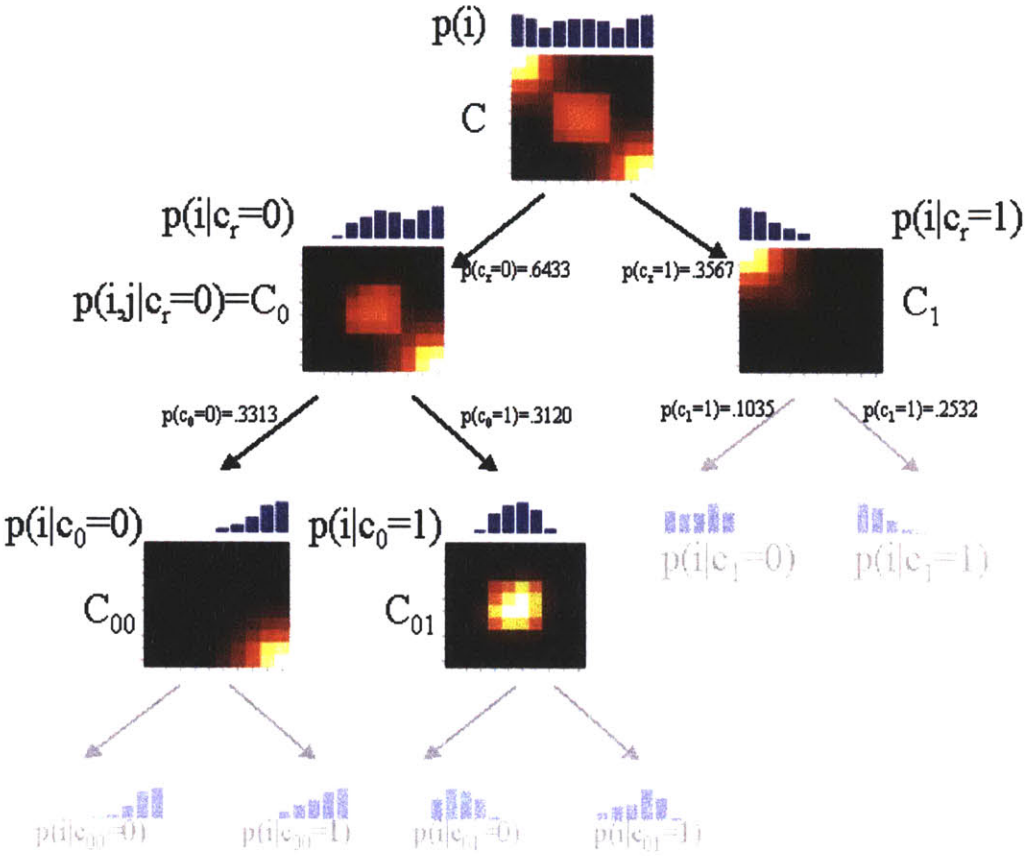


Figure 4-8: This figure shows the hierarchical clustering of the "three person" experiment. The first branch separates observations from the first two individuals from the third. The second branch on the left separates the first and second person. Repeated trails result in this solution or a solution with person two and person three sharing a node, but never person one and three sharing a node.

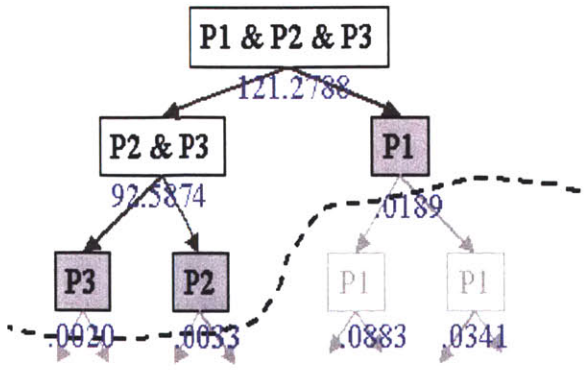


Figure 4-9: This figure shows values of d for successive binary segmentations.

the leaf nodes of any pruned version of the binary tree describe a set of latent class models whose weighted pmfs approximately sum to the cumulative observation pmf and whose co-occurrence statistics sum to the observed co-occurrence statistics.

We prune the tree after any node whose children's co-occurrences are likely to have been drawn from that node's co-occurrences. There are two significant cases where this is true. If the weight of either child is very small, the second child will be very similar to the parent, hence, the co-occurrences will be very similar. If the pmfs of both children are very similar to the pmf of the parent, their co-occurrences will be very similar regardless of the weight on the two children. In either of these cases, further branches are very unlikely to produce interesting results.

The similarity can be quantified by the kl-divergence between the parent co-occurrence matrix and the weighted sum of the childrens' co-occurrence matrices.

$$d = \sum_i \sum_j C_{i,j}^{parent} \log \frac{C_{i,j}^{parent}}{C_{i,j}^{children}} \tag{4.33}$$

where $C^{children}$ is the weighted co-occurrence of the two child nodes.

$$C^{children} = p(c_0)C^{c_0} + p(c_1)C^{c_1} \tag{4.34}$$

Figure 4-9 shows the values of d corresponding to the segmentation in Figure 4-8 overlaid onto the pruned version of the tree. Figure 4-16 shows an example of a complete tree before pruning. The next section illustrates how a set of latent class models can be used to classify MOSs robustly despite potentially ambiguous observations.

4.2.4 Classifying an MOS

Classifying an MOS involves determining which class is more likely to have produced the MOS. The likelihood of a class given an MOS is

$$p(c|M^u) = \frac{p(M^u|c)p(c)}{p(M^u)} \tag{4.35}$$

$$= \frac{p(M^u|c)p(c)}{\sum_d p(M^u|d)p(d)} \tag{4.36}$$

$$= \frac{p(c)\prod_{x\in M^u} p(x|c)}{\sum_d p(d)\prod_{x\in M^u} p(x|d)} \tag{4.37}$$

This is the maximum likelihood latent class for M^u is

$$c_u = \underset{c}{\text{argmax}}\, p(c|M^u) \tag{4.38}$$

$$= \underset{c}{\text{argmax}} \frac{p(c)\prod_{x\in M^u} p(x|c)}{\sum_d p(d)\prod_{x\in M^u} p(x|d)} \tag{4.39}$$

$$= \underset{c}{\text{argmax}}\, p(c) \prod_{x\in M^u} p(x|c) \tag{4.40}$$

$$= \underset{c}{\text{argmax}} \log\left(p(c)\right) \sum_{x\in M^u} \log\left(p(x|c)\right) \tag{4.41}$$

$$= \underset{c}{\text{argmax}} \log\left(p(c)\right) \sum_{x\in M^u} N(x = O_i) \log\left(p(i|c)\right) \tag{4.42}$$

$$= \underset{c}{\text{argmax}} \log\left(p(c)\right) E_{O_i\in M^u}[\log p(i|c)] \tag{4.43}$$

where $N(x = O_i)$ is number of observations in M^u that are equivalent to O_i and E is an expectation.

Figure 4-10 shows two classes, c_0 and c_1, which output sets of observations from one to four. The first class is equally likely to produce observations from one through three. The second class is equally likely to produce observations from two through four. Both classes are nearly equally likely to occur. Thus, the production parameters are

$$p(i|c_0) = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0\}\ p(c_0) = .5 + \epsilon, \tag{4.44}$$

$$p(i|c_1) = \{0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}\ p(c_1) = .5 - \epsilon. \tag{4.45}$$

Using the maximum likelihood classification policy, an MOS with a single observation would be classified as class 0 if the observation was in $\{1, 2, 3\}$ and class 1 otherwise. This results in $\frac{1}{3}$ classification error rate. As the size of the MOS increases, the classification error decreases because a larger set of observations is more likely to contain an unambiguous example.

Given a perfect discriminant model for single observations, one could optimally

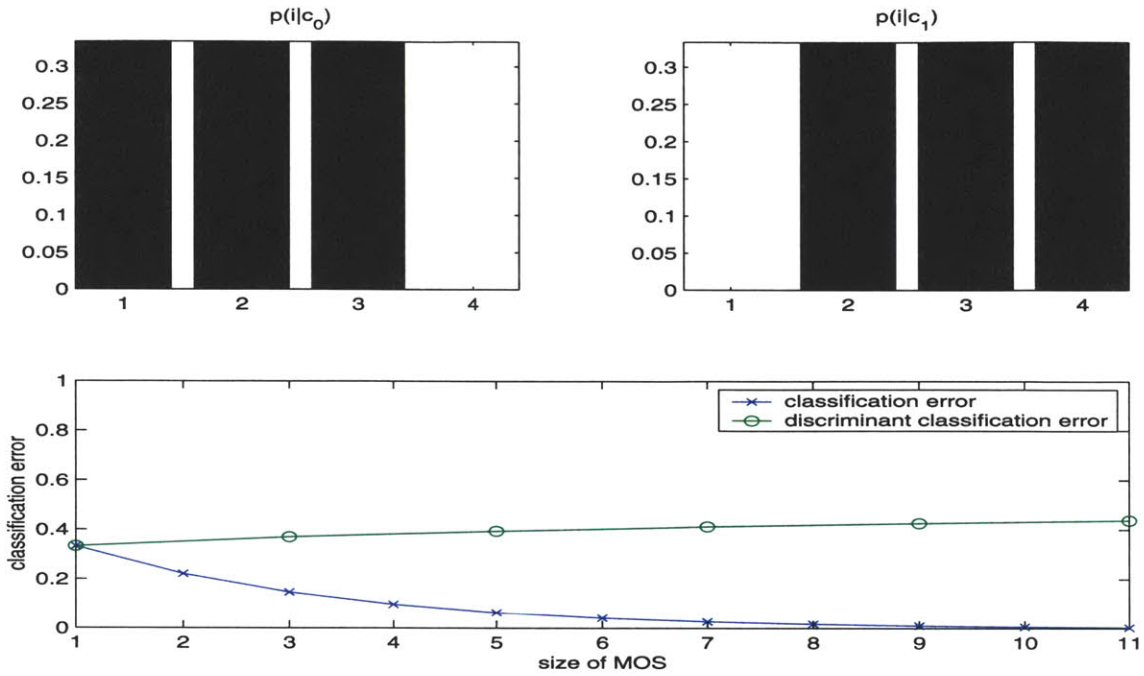


Figure 4-10: This figure shows the classification error for a simple example for clustering MOSs of different sizes. Given more observations, the classification error decreases unless no model of uncertainty is available when classifying observations, in which case the error increases.

classify single examples. But in our simple example, the classification error will actually increase with more observations without some measure of confidence on that discriminant measure because approximately two thirds of the observations from class 1 will be misclassified.

Figure 4-10 shows the error for these two cases. This example underscores the importance of a probabilistic class model as opposed to a discriminant model of the observations. A probabilistic model will always perform as well as a discriminant model, but it will perform significantly better with larger MOSs when some observations are ambiguous. As stated earlier, this is often the case in perceptual data. Alternative pairwise clustering mechanisms are available which do not suffer from local minima, but they make hard assignments of observations to clusters.

4.2.5 Discrete space considerations

Thus far, we have used explicitly chosen examples to illustrate MOL. This section explores the assumptions and limitations of MOL.

Estimation, storage, and computational requirements

The number of co-occurrences that are required to estimate the co-occurrence matrix increases as the square of the size of the observation space, K. In our simple example

K was ten, but for most interesting problems K is considerably large. The number of co-occurrences from an MOS increases with square of the size of the MOSs. Thus, the number of MOSs required to estimate a co-occurrence matrix is proportional to $\frac{K^2}{S^2}$. The storage requirements as well as the computational requirements in estimating the latent class models are also $O(K^2)$. These factors will be important considerations in the continuous MOL section.

Embedding of discrete observations has no effect on results

Note that there is no assumption about the embedding of the discrete pmf. The "three person" experiment could involve three people who prefer even numbers, odd numbers, and primes respectively. The people could also be asked to give their favorite animals as long as their samples could be described by a pmf over dogs, cats, cows, horses, etc.

Convergence

Convergence of this type of minimization procedure has been previously investigated [34]. But there are no guarantees of convergence to a globally optimal solution.

Non-IID data

Our experiments involved independent and identically distributed samples. This is not the case for most perceptual observations. For instance, a person tracked for only two frames will result in an MOS with two similar silhouettes rather than two completely random person silhouettes. A person tracked for a single frame in two different cameras may produce two independent silhouettes, but that pair of silhouette is more likely to reoccur in that pair of cameras than a random pair of silhouettes. In a particular scene, each vehicle may only show variation across a small set of angles.

Thus, one class of object can result in more than one cluster. Fortunately, these clusters usually have a significant probability of producing some of the same observations and will often be clustered in similar branches of the hierarchy. A case where this would not be true is if a scene contained two roads at different angles and no vehicles were observed moving from one road to another. In this case, there is no direct visual evidence that the vehicles on one road are similar to vehicles on the other road. After correcting for a planar projection, the object size and speeds may be closely related, but in general this situation would require user-provided supervision.

Identifiability

Given infinite data, if our estimated latent class parameters exactly match the parameters of the production model, the error will be minimized. Unfortunately, the inverse is not true because multiple latent class models may result in identical co-occurrence matrices. This is obvious when considering multiple classes that have identical observation pmfs.

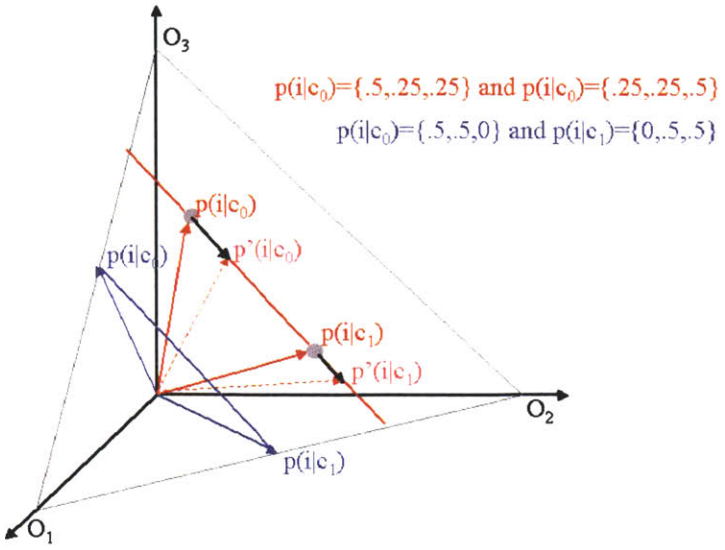


Figure 4-11: This figure shows two two-class examples outputting over three output observations. The blue example produces a unique co-occurrence matrix. The red example shows two sets of model parameters that produce the same co-occurrence.

As a simple example, consider a coin flip experiment. Imagine taking every coin from a coin press and flipping it twice. Given an infinite number of coin flips, our co-occurrence matrix will converge to C_{coin} representing the joint distribution of coin flip pairs.

Because there are only two output observations (heads and tails), the co-occurrence matrix has four elements with three free parameters because it sums to one. Even with a single type of coin, little can be determined about the system. If there is only one type of coin, the joint distribution should be completely independent (the joint is the exact product of the marginal distributions). If the joint distribution is not independent (given infinite samples), there must be at least two types of coins with different biases. The only identifiable case with two coins and two observations is where one class produces only the first observation and the second produces only the second. In all other cases, multiple latent class models produce exactly the same co-occurrence matrices and additional knowledge is required. For instance, if the weight of the two classes of coins is known, it is possible to estimate the latent class parameters for the system.

Figure 4-11 shows two two-class models in a three dimensional output space. The pmfs for two generative models are shown by pairs of blue and red arrows. The blue model produces a unique co-occurrence matrix. The red model produces an ambiguous co-occurrence matrix. By altering both $p(i|c_0)$ and $p(i|c_1)$ by the proper amount in any direction in the subspace defined by the two points and altering the class weights, an identical co-occurrence will result.

In multi-class problems, any set of latent class pmfs that can all be altered in a direction within the subspace defined by the same set of pmfs without moving any

pmf off the probability simplex will result in a non-unique co-occurrence matrix. The blue example is unique because moving both points in either direction would force one point off the probability simplex. Obviously, any production model in which one class pmf is not necessary in defining the minimal convex region containing all pmfs does not produce a unique co-occurrence matrix. A sufficient condition for production model identifiability is that each latent class have one observation that it alone produces.

4.3 MOL in continuous observation spaces

If the observations are continuous, it is not possible to estimate co-occurrence statistics because two identical observations may never occur. This may also be the case in extremely large observation spaces. Also, the amount of storage required, the amount of data necessary to estimate a co-occurrence matrix, and the computational requirements increases with the square of the size of the observation space. In these cases, it is necessary to determine a discrete representation for the MOSs by grouping local observations together.

4.3.1 Uniform tiling of the observation space

A trivial way to accomplish this is to tile the space of observations into equal sized regions. All observations in each region are treated as the same discrete observation. If the "three person" experiment was altered to allow people to provide continuous values, it would be necessary to quantize the space in order to collect co-occurrence statistics. To quantize this range to the same values as in the previous experiment $(1, 2, 3, ...10)$, we can introduce those values as codebook prototypes. Each continuous observation is represented by the index of the nearest prototype. The discrete probability mass function (pmf) given person c is

$$p(x|c) = \int_{x-.5}^{x+.5} p(y|c)dy \tag{4.46}$$

where x can take on the values of integers from one to ten, $p(y|c)$ is the continuous probability density function (pdf) of class c (person c). If the approximation to the continuous density resulted in the pmfs shown in Figure 4-3, this experiment is exactly equivalent to the previous experiment. In the final section, we will show that our method is invariant to this tiling in as much as $p(x|c) \simeq p(y|c)$.

Unfortunately, simply tiling high-dimensional input spaces will often result in poor pmf approximations to pdfs unless the regions are very small. But, if the tiling is fine, many of the regions of the space will contain few observations. Regions that contain few samples will not produce reliable co-occurrence statistics and will result in an inefficient encoding. Representing co-occurrence statistics using an inefficient tiling increases the storage and computation requirements.

For these reasons, using vector quantization to determine a representative set of

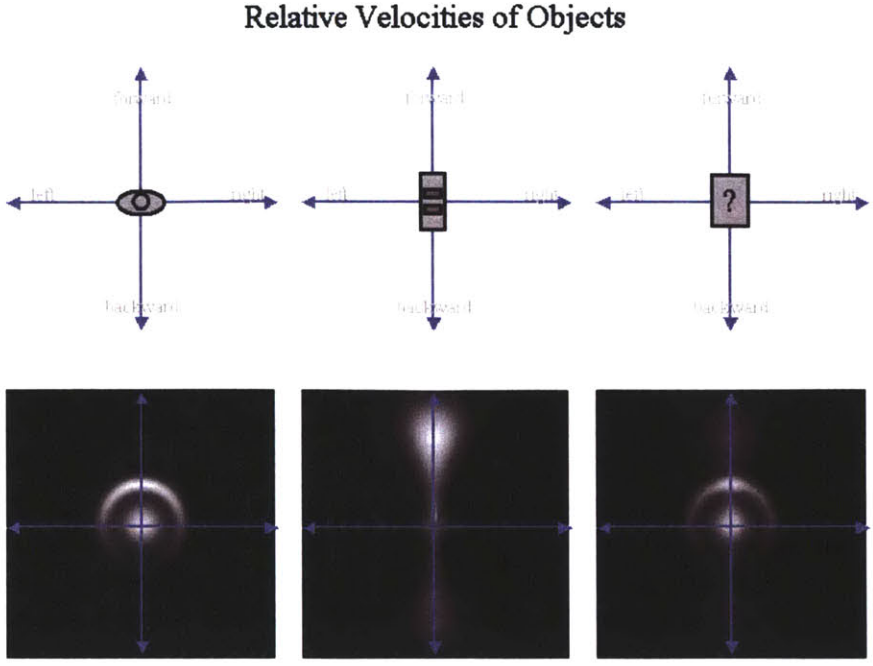


Figure 4-12: *This figure shows artificially produced histograms of velocities of pedestrians, vehicles, and an unknown mixture of the two velocities profiles. The vertical axis denotes the velocity in the direction of movement. Positive values are forward and negative values are backward. The horizontal axis denotes the velocity in the direction perpendicular to the direction of movement.*

prototypes tends to increase performance. This enforces the constraint that each region contain approximately the same amount of data. In the Subsection 4.3.3, we introduce a class-conditional tiling which determines a set of regions which represent the data *and* contain mostly observations from one class.

4.3.2 Codebook generation

The goal of codebook generation is to determine a set of K prototype observations which can be used to efficiently represent a set of continuous observations. Given this set of prototype observations, each observation in an MOS can be represented by the index of the most similar prototype. Thus the problem is reduced to the previously covered discrete MOL.

The codebook must be kept reasonably small because of computational, storage, and estimation limitations. The codebook should be large to increase the expressive power of the resulting latent class models. The codebook should be representative of the observations so elements of the codebook are not wasted representing observations that rarely occur.

In our previous example, an MOS of continuous values from one to ten (8.2, 1.8, 1.2, 4.0138, ..., 2.3) would be represented as $(8, 2, 1, 4, ..., 2)$. The tiling could be made

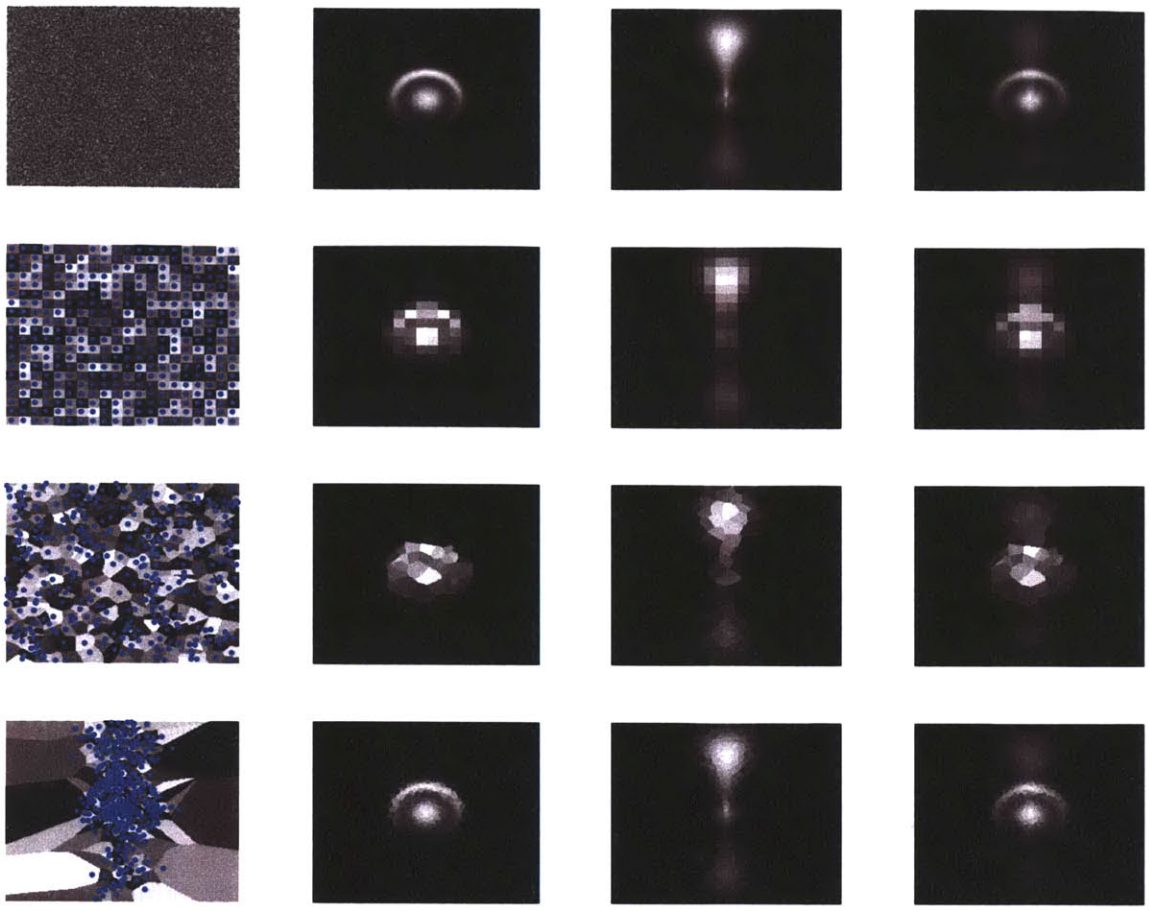


Figure 4-13: *This figure shows three approximations of pdfs using the same number of prototypes. The first line shows: the tiling of the space and the original pdfs for pedestrians, vehicles, and a mixture of the two classes. The following three lines show approximations using 400 prototypes using a uniform distribution, a random distribution, and a distribution that is representative of the data.*

finer to make the discrete probability mass function (pmf) a better estimate of the continuous probability distribution function (pdf). But if it is made too fine, the co-occurrence matrix will be extremely large and computation of the latent class models will be prohibitive. Also, as the size of the co-occurrence matrix increases so does the relational supervision requirements.

Figure 4-12 shows a less trivial example. Relative velocity profiles for two types of objects are shown–people and vehicles. From the velocity probability distribution functions (pdfs), it is apparent that vehicles move faster than pedestrians but pedestrians are more likely to change their direction and less likely to walk backwards. Both classes are likely to stop, although people exhibit more noise when stopped because they are less stationary than vehicles.

There are many ways to produce a codebook of prototypes for this two dimensional space. Figure 4-13 shows the approximations resulting from three distributions of 400

	random	uniform	representative
L1-error	$.2395 \pm .0059$	.2159	$.1914 \pm .0084$
L2-error	$9.5598x10^{-6}$	$8.2583x10^{-6}$	$5.0707x10^{-6}$
kl-divergence	.0937	.0543	.3829

Table 4.1: *This table shows the pmf approximation error for the three sets of prototypes shown in 4-13.*

prototypes: a uniform distribution, a random distribution, and a distribution that is representative of the data. Table 4.1 shows the errors of the pmf approximations to the true probability distributions.

Assignment to a uniform distribution of prototypes results in a uniform tiling of the space. It can be performed using a simple hash function. The error is always larger using 400 random prototypes as opposed to distributing the prototypes uniformly in a grid. While kl-divergence is minimized for the uniformly distributed prototypes, L1-error and L2-error are minimized for the representative prototypes. Further, as shown in Subsection 4.4 the representative prototypes enable better latent class estimation.

Online codebook generation

There are many methods of developing codebooks of prototypes that are representative of the data (see [15] for a discussion). For the quantity of data and the number of prototypes available to our system, an off-line method, such as K-means, is not feasible on today's hardware. To allow our system to handle large amounts of data, we have chosen to use on-line vector quantization.

The simplest method of on-line vector quantization is to initialize the codebook randomly with K prototypes centered at existing data points. Then, take single data points, find the closest prototype in the codebook, and adapt that prototype towards the data point using a learning factor, α. This process is repeated for millions of data points as the α value is slowly decreased until the prototypes are stable and represent an approximately equal amount of data. The continuous observation spaces we encountered did not require complex annealing strategies.

We occasionally encountered an initialization problem. Prototypes seeded on outlying data points may be stranded representing only that data point. We circumvented this problem with a method similar to Johnson and Hogg[27] which enforces that each prototype represent the same amount of data. Over time, stranded data points account for larger regions of the input space until they represent new data points. The prototypes are then adapted towards the new data points until they represent as much data as all the other points. This restriction can be softened in situation where there are very common observations.

Once a codebook is generated, it is used to represent continuous data points, i.e., continuous observations are represented by the indices of the nearest prototype. Given the desired size of the codebook, the goal of quantizing is to determine a set of prototypes which best represent the dataset. Our results were produced with codebooks of 400 prototypes. More complex spaces (e.g. color image space) would

necessitate either more prototypes or more complex prototypes.

In high dimensional observation spaces with complex class pdfs, it may be difficult to create an effective codebook of prototypes. If all the representations in the codebook are equally likely to result from all the underlying classes, the resulting pmf approximations will be equivalent. For example, if none of the representations in your codebook is more likely to result from a person than a vehicle, there will be no possibility of using those representations to differentiate people and vehicles.

While this may seem unsettling, we are encouraged by our ability to generate large codebooks. Large codebooks are usually troublesome because as the size of the codebook, K, increases, the amount of data needed for effective codebook generation increases on the order of K. Also, the amount of data needed for estimating the co-occurrence statistics increases on the order of K^2. Since our system automatically collects and processes data, we have hundreds of gigabytes of tracking data available. And, our method converges as the amount of data increases rather than suffering from over-fitting, because more data will result in a better estimate of the co-occurrence matrix.

An area of high data point density may accumulate a large portion of the prototypes, leaving few prototypes for the rest of the input space. In some cases, it may be desirable to have a large number of prototypes in the high-density areas because those regions may be the most ambiguous regions of the input space (e.g. traffic at an intersection). In other cases, the areas of high density may arise from uninteresting, repetitive input data (e.g. scene clutter) and there is usually no benefit to wasting a large portion of your prototypes in that region. We currently filter most of the MOSs whose observations were captured in less than a few seconds. This filters most of the repetitive motions in the scene before MOL begins.

4.3.3 Associative mixture of Gaussians (AMG)

In the following section, we have illustrate that classification performance generally improves when using prototypes that are representative of the observations. Classification performance is often significantly better when using a set of prototypes to represent each class. Figure 4-14 shows an example where this is the case.

If the latent class of each MOS was known and sets of Gaussians were assigned to represent each class, we could maximize the likelihood of the data under a set of independent mixtures of Gaussians.

$$P(x) = \sum_{j=1}^{M} p(x|j)p(j)\delta_{i,j} \tag{4.47}$$

where $\delta_{i,j} = 1$ if observation i and Gaussian j are from the same class. If the class assignments of both the observations and the Gaussians were known, this would simply reduce to a mixture of Gaussians estimation for each class using only the observations and the Gaussians assigned to that class. Unfortunately, neither the observation or Gaussian class is known, but as shown earlier in this chapter it is possible to estimate both.

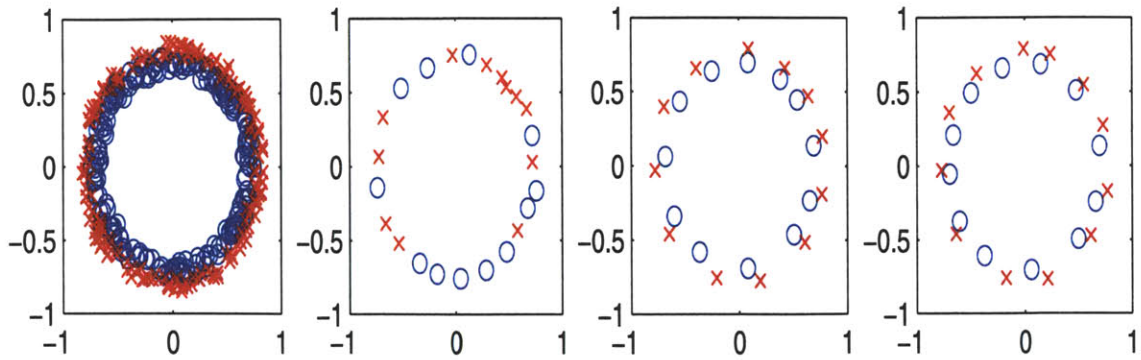


Figure 4-14: This figure shows two concentric circle distributions of slightly different radius (a). Using standard codebook generation (b), each prototype represents significant portions of both classes of observations. Prototypes are labelled with their most likely latent class, but most of these prototypes are nearly equally likely under each class. This results in poor latent class estimation and poor classification. Using a separate set of prototypes for each latent class (c) results in significantly better classification. (d) shows the result of Associative Mixtures of Gaussians.

We have illustrated that it is possible to determine the latent class probabilities given the number of classes, an assignment of observations to prototypes, and co-occurrences of those observations. This allows us to determine an expected value of $\delta_{i,j}$ as

$$E[\delta_{i,j}] = \sum_c p(c|x,g) = \sum_c p(c|x)p(c|g) \tag{4.48}$$

This expectation is one or zero if there is no uncertainty of the class of the observations or the Gaussians. Using this estimate of $\delta_{i,j}$, it is possible to do a maximum likelihood estimation of the parameters of the likelihood given in Equation 4.47. After re-estimating the parameters of each of the Gaussians, some observations may be assigned to different Gaussians, thus the co-occurrences must be re-estimated and the latent classes must be determined again. Because a single iteration of estimation is not likely to change many of the assignments, it is not necessary to start from random conditions each time or to wait until full convergence. Both the estimation of the latent class and the estimation of the mixture parameters can continue in parallel.

By Bayes Rule,

$$P(c|g) = \frac{p(g|c)p(c)}{p(g)} \tag{4.49}$$

$$= \frac{p(g|c)p(c)}{\sum_c p(g|c)p(c)} \tag{4.50}$$

and $p(c|x)$ is the likelihood of the class given the current observations MOS given in Equation 4.37.

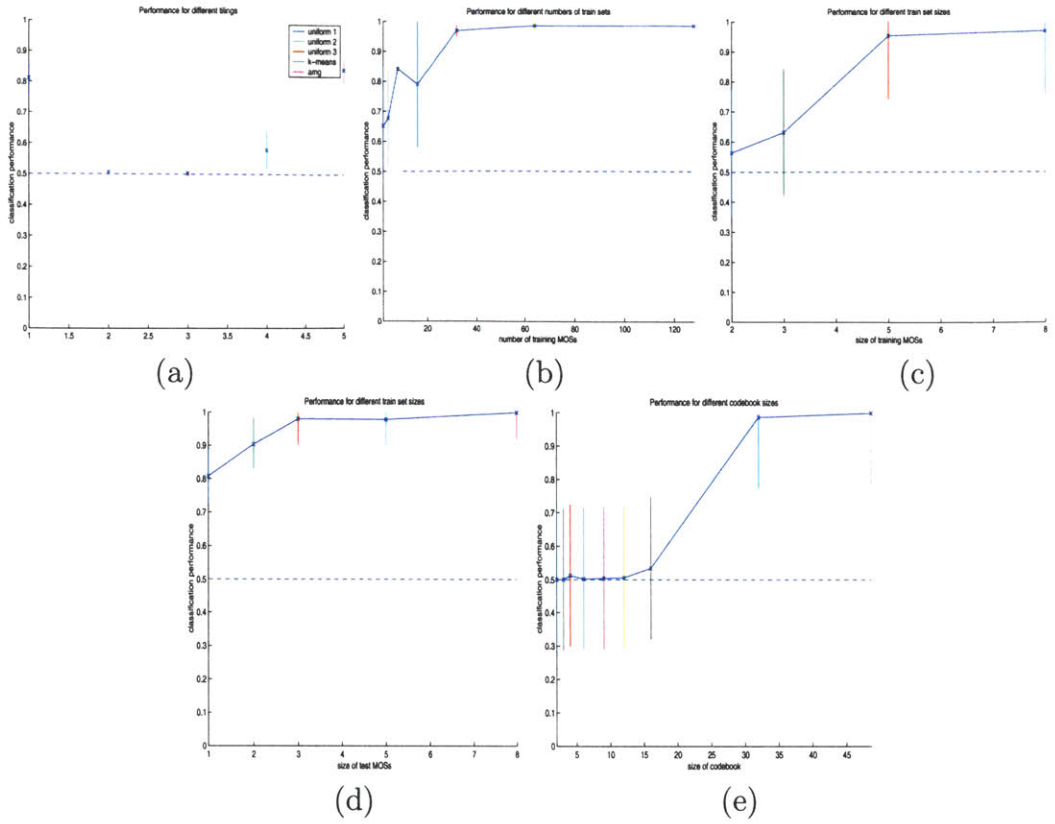


Figure 4-15: This figure shows the effect on classification performance of different tiling methods (a), number of training MOSs (b), training MOS size (c), test MOS size (d), and number of prototypes in codebook (e). Each graph shows the classification error for a binary classification problem assuming the clusters were labeled with the most likely class. Large error bars resulted from convergence variability. Each experiment is discussed in Section 4.4.

4.4 Performance factors

This section discusses the many factors that affect the performance of Multiple Observation Learning (MOL) including type of tiling, number of training MOSs, size of training MOSs, size of test MOSs, size of codebook, and overlap of latent class pdfs. Each of these will be discussed in the following subsections using the previous (two-ring) latent class example and other simple examples to illustrate their effect.

Figure 4-15(a) shows classification results with different values for each of the above factors. Except where noted, these experiments involved codebooks containing 25 prototypes. There were 20 MOSs that contained 20 observations used to train the system. The test classification was performed on 500 MOSs with 4 observation. The Associative Mixtures of Gaussians tiling was used except where noted. In all cases, two latent classes were assumed.

Type of tiling

Figure 4-15(a) shows the performance of the MOL system using five different tilings. The first three tilings are uniform tilings of 25 prototype observations in which the data falls into approximately 8%, 36%, and 96% of the tiles. This illustrates the effect of sparsity of the data in the space. If the observations are sparse, the uniform tiling may not have enough prototypes in the regions near the observation to have the discriminability necessary to differentiate classes. This becomes a very important issue in high dimensional spaces.

The last two tilings are straight K-means prototypes and AMG prototypes. As shown in figure 4-14, a K-means prototype will tend to represent nearly the same number of observations from both classes whereas the AMG prototypes tend to represent more observations from a single latent class. This is results in improved classification performance. For every experiment we performed with more than one or two dimensional observations, uniform tilings were too inefficient. In most cases, AMG is not required which is fortunate because it must be performed in batch with all observations being accessed in each computation cycle. We use a set of representative prototypes in most cases.

Number of training MOSs

As the number of training MOSs increase, performance tends to increase and the variability in the results tends to decrease. With very few MOSs, the class-conditional pmfs are not representative of the latent class pmf and therefore the training classification is poor. Figure 4-15(b) shows this tendency.

Size of training MOSs

As the size of the training MOSs increases, the estimates of the latent class pmfs becomes more reliable. This also affects the estimate of the latent class of the MOSs and can result in better AMG results. Figure 4-15(c) shows this tendency.

Size of test MOSs

As shown earlier in our simple example, the classification performance increases with the size of the MOS.

Size of codebook

Without enough prototypes in the codebook, prototypes tend to represent observations from both latent classes. Thus the co-occurrence matrices do not specify a unique generative model and latent class weights can be poorly estimated. This can cause one latent class model to have a significantly higher prior and result in MOSs from both classes being classified as the same class and, thus, near random classification error. After enough prototypes are alloted, the performance improves. In our two-ring example, performance was significantly better if at least 12 prototypes

were used. The number of prototypes required increases with the complexity and proximity of the latent classes.

Overlap of latent class pdfs

As was discussed in Subsection 4.2.5, the latent classes do not have to be separable but each latent class has to represent one distinct set of observations in order to reliably estimate the latent class parameters.

4.5 Results

We assume that the attention and correspondence systems will produce sets of observations of the same objects, or MOSs. For example, a person tracked through the scene for N frames will result in a set of N images, N binary silhouettes, N positions, N velocities, etc. Because our correspondence system associated a much higher cost to false positives than false negatives, our MOSs will rarely contain observations of multiple objects. Some sequences will be broken when correspondence is uncertain to avoid the possibility of creating false correspondences. So, except in rare cases, every observation in an MOS should correspond to the same object in the world. No two observations will be exactly the same. Each observation shows the object at a different time or from a different angle.

The following two examples involve creating a classification hierarchy using the same number of prototypes, the same learning parameters, and the same sequences produced by our tracking system. The only difference is that they use different portions of the observation description. The first example classifies activity based on a 5-tuple (image position, speed, direction, and size). The second example classifies shape based on a 1024-tuple (32x32 binary silhouettes).

4.5.1 Classifying activities

This example clusters objects based on a representation of their position, speed, direction and size (x,y,dx,dy,s). First, four hundred representative prototypes are determined. Each prototype represents the objects of a particular size that are seen in a particular area of a scene moving in a particular direction. Co-occurrences are accumulated using 24 hours of MOSs from that scene. Finally, the universal pmf (the true pmf of the entire set of sequences) is probabilistically broken into two pmfs. The process is repeated to produce a binary tree of height four detailed in Figure 4-16. Figure 4-17 shows the history of one particular day.

Note that the scene contains a road with adjacent parking spots and a path through the grass near the loading bay of our building. The binary tree shows accumulated motion templates for each node of the tree. The first break separates traffic moving in one direction around the building and traffic moving in the other direction, because objects in this scene did not generally change their direction. The second break for both branches separates traffic on the road and traffic on the path. While

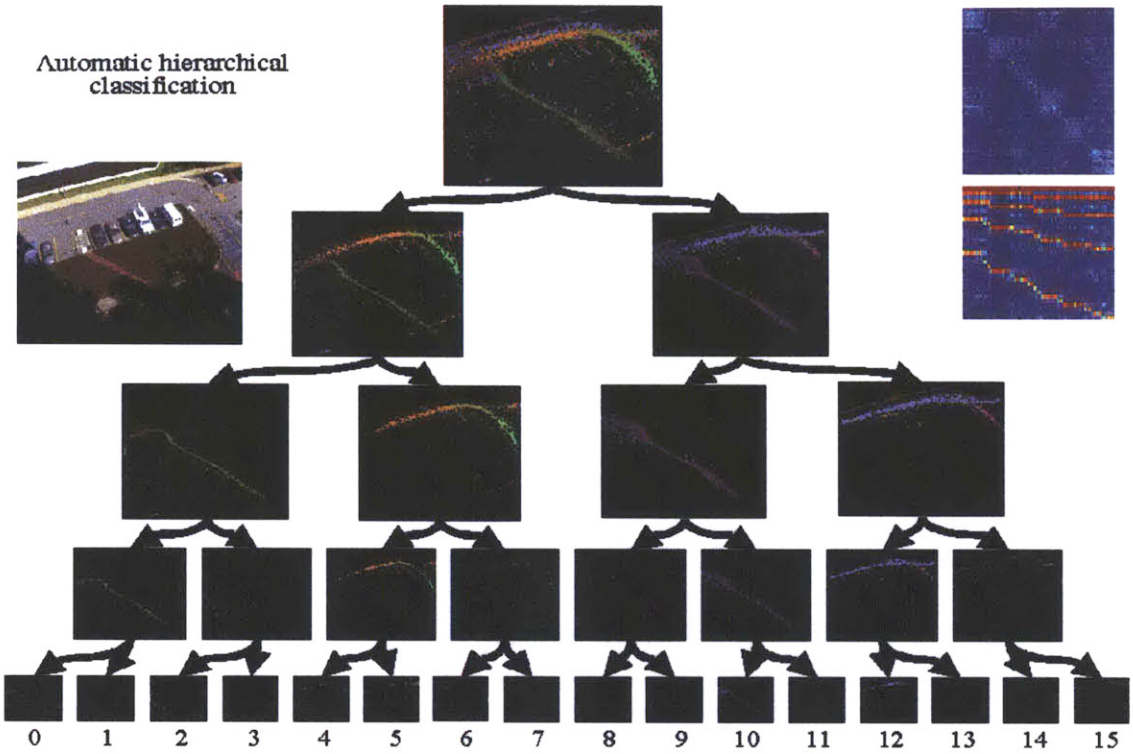


Figure 4-16: *This figure shows an image of the scene(upper left), the classification hierarchy(center), and the co-occurrence matrix and normalized pmfs(upper right) for each element of the tree. The scene contains a road with adjacent parking spots and a path through the grass near the loading bay of our building. The binary tree shows accumulated motion templates for each node of the tree. And the co-occurrence matrix and normalized pmfs show which prototypes occurred within the same sequences and the probability distributions for each node in the tree (ordered breadth-first). The final level of the tree specific classes including: pedestrians on the path (one class in each direction); pedestrians and lawn-mowers on the lawn; activity near the loading dock. cars; trucks; etc. These classes can be viewed in a Java 1.1 compatible browser at:* `http://www.ai.mit.edu/projects/vsam/Classification/Cclasses/` . *Note: the columns and rows of the co-occurrence matrix have been ordered to make some of its structure more apparent.*

Classification counts for 9/21/98

	Level 2		Level 3				Level 5															
Time	node 0	node 1	00	01	10	11	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
12am-1am	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1am-2am	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2am-3am	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3am-4am	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4am-5am	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5am-6am	1	3	0	1	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
6am-7am	6	29	3	3	10	19	3	0	0	0	3	0	0	0	0	0	10	0	17	0	0	2
7am-8am	18	76	4	14	27	49	3	1	3	0	6	3	3	2	0	1	25	1	44	2	1	2
8am-9am	32	78	8	24	32	46	5	0	2	1	3	11	8	2	0	1	28	3	34	3	3	6
9am-10am	72	43	26	46	19	24	17	1	7	1	14	12	16	4	1	0	15	3	9	0	5	10
10am-11am	31	19	10	21	8	13	8	0	2	0	6	4	11	0	0	1	4	1	4	2	3	4
11am-12pm	38	21	19	19	9	12	14	1	2	2	6	2	6	5	3	0	5	1	6	1	2	3
12pm-1pm	60	62	36	24	38	24	34	1	1	0	11	6	7	0	0	0	34	4	11	0	5	8
1pm-2pm	20	32	10	10	13	19	7	1	2	0	5	1	3	1	0	0	12	1	11	2	4	2
2pm-3pm	23	27	6	17	15	12	4	0	0	2	9	3	3	2	1	0	14	0	5	0	1	6
3pm-4pm	42	27	15	27	13	14	11	0	3	1	21	3	2	1	0	3	7	3	6	0	5	3
4pm-5pm	60	30	28	32	12	18	24	0	4	0	28	0	3	1	0	0	12	0	9	0	6	3
5pm-6pm	67	33	27	40	16	17	20	1	5	1	32	3	4	1	0	0	14	2	5	4	3	5
6pm-7pm	18	14	3	15	9	6	2	0	1	0	13	1	1	0	0	0	9	0	2	0	1	3
7pm-8pm	8	4	0	8	0	4	0	0	0	0	0	0	3	0	0	0	0	0	0	0	2	1
8pm-9pm	4	10	0	4	0	10	0	0	0	0	0	0	4	0	0	0	0	0	1	0	5	4
9pm-10pm	4	0	2	2	0	0	0	0	2	0	0	1	1	0	0	0	0	0	0	0	0	0
10pm-11pm	4	5	0	4	1	4	0	0	0	0	1	2	1	0	0	0	0	0	1	0	2	1
11pm-12pm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	East-bound	West-bound																				

See http://www.ai.mit.edu/projects/vsam/ to view these activity clusters in a Java applet

Figure 4-17: *This figure shows how many of the activities were detected on a particular day. The first two columns correspond to the initial branch. The following four columns correspond to the next level of the binary classification tree. The last 8 columns are the leaf nodes of the classification tree. Below some of the columns the primary type of activity for that node is listed. Morning rush hour is highlighted in green(light gray) and shows traffic moving mostly in one direction. The lunch-time pedestrian traffic is highlighted in red(gray). The evening rush hour is highlighted in blue(dark gray) and shows more movement in the opposite direction as the morning rush hour.*

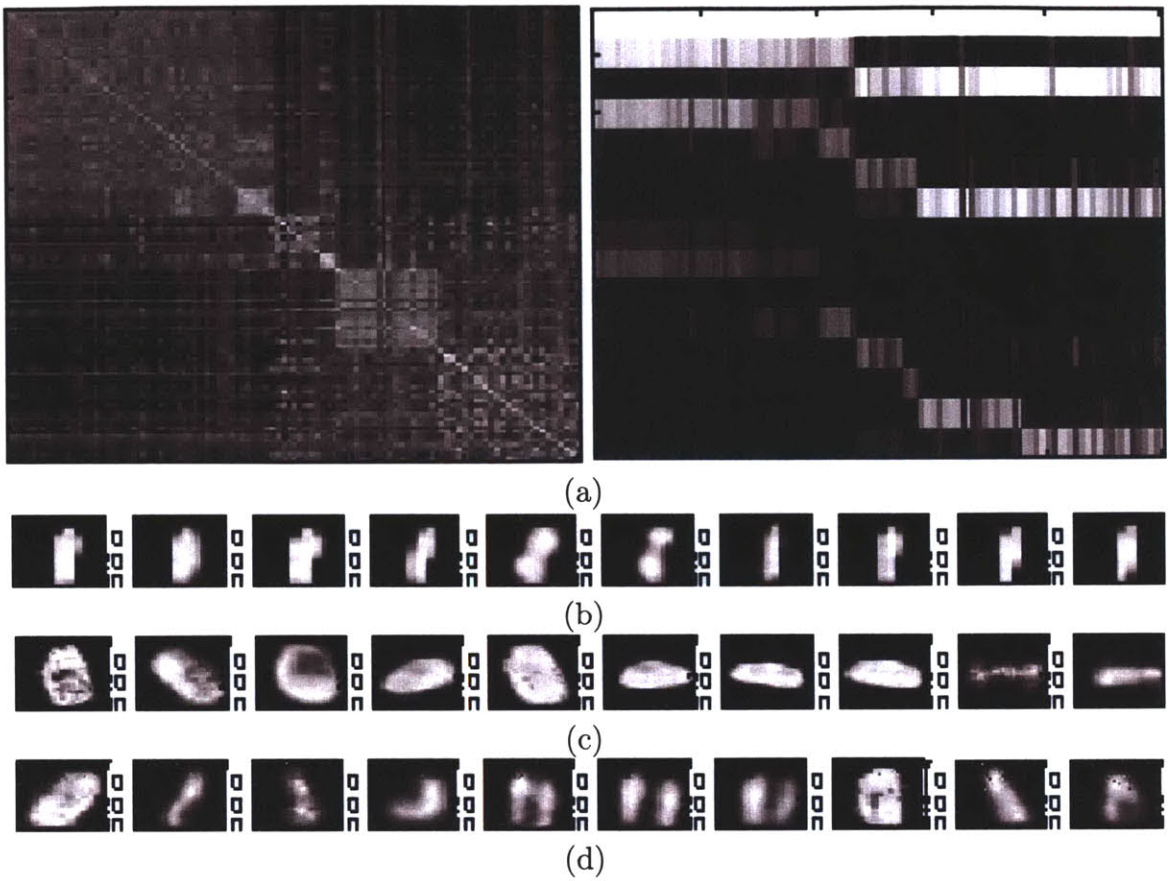


Figure 4-18: *(a) shows the co-occurrence matrix and resulting pmfs. Some of the prototypes from the person class(b), vehicle class(c), and some prototypes which were significantly ambiguous(d). In C, the upper left corresponds to silhouettes of people and the lower right corresponds to silhouettes of vehicles. The vehicles show less statistical independence because vehicles in this particular scene were only scene as they passed through particular orientations. If the scene contained vehicles driving in circles, the corresponding prototypes would exhibit more independence. Note: the co-occurrence matrix has been ordered to make some of its structure more apparent.*

there are some prototype states characteristic of both activities, these two activities were significantly different and accounted for a significant amount of the data. Further bifurcations result in classes for: pedestrians on the path; pedestrians and lawn-mowers on the lawn; activity near the loading dock. cars; trucks; etc. These classes can be viewed in a Java 1.1 compatible browser at:

http://www.ai.mit.edu/projects/vsam/Classification/Cclasses/.

4.5.2 Classifying motion silhouettes

This example clusters MOSs based on their observation silhouettes. Because of the dimensionality and sparsity of binary silhouette images, 400 representative prototypes were estimated that correspond to people, vehicles, and various other moving objects

or artifacts. After co-occurrence of these prototypes in the MOSs are accumulated, the hierarchical latent class model is estimated.

The first branch of the hierarchical clustering broke the silhouettes into two relatively discrete classes, people and vehicles. Some of the more blurry prototypes remained ambiguous because they matched both vehicles and people. These prototypes were shared between the two classes. Figure 4-18 shows the co-occurrence matrix, the pmfs, and some examples of prototypes from both classes.

Figure 4-19 shows classification of a day of silhouette sequences. After pruning, the resulting classifier first separated vehicles as they were decisively different from the other silhouettes. This means that while vehicles appeared at many different angles within their sequences, few sequences contained both vehicles and people. The next break was individual pedestrians. The last break separated groups of pedestrians from clutter and lighting effects. Figure 4-19 shows the distribution of events over a 24 hour period, highlighting the changes in density of pedestrian and vehicular traffic as a function of time.

The daily activity histograms show some interesting facts. The highest occurrences of people and cars was in the morning and evening as expected. Groups of people tended to occur most shortly after noon. The clutter was primarily trees, garbage, and lighting effects on the side of buildings. The histogram and images show that it was a very windy morning and the lighting effects occurred near dusk.

4.5.3 MOS classification summary

The previous two examples have shown robust, hierarchical clustering based on shape and activity. Both examples take advantage of the co-occurrence of observations in multiple observation sets (MOSs) to estimate probabilistic latent class models. The next two sections show two additional applications of the MOL framework. First, by modeling co-occurrence of pixels in regions of regularity in pedestrian images, we can hierarchically, probabilistically group pixels into regions of regularity closely corresponding to parts often used for description of individual's appearance. Second, using spatial co-occurrence of pixel values in magnetic resonance images, we can estimate latent tissue class models.

4.6 The Similarity Template (ST)

Detection, alignment, and recognition in color images are often approached with completely different representations of image patches. For detection and alignment of a class of objects, a representation is sought that is invariant to the color of a particular object (e.g., edge templates, gray-scale Haar wavelets, etc.). In contrast, for recognition of a particular instance, often the colors of particular regions are extremely important in differentiating instances.

An illustrative example is detecting pedestrians as opposed to differentiating pedestrians. The class of pedestrians can be described as a configuration of a few regions of regularity surrounded by other regions of regularity. For pedestrians, these

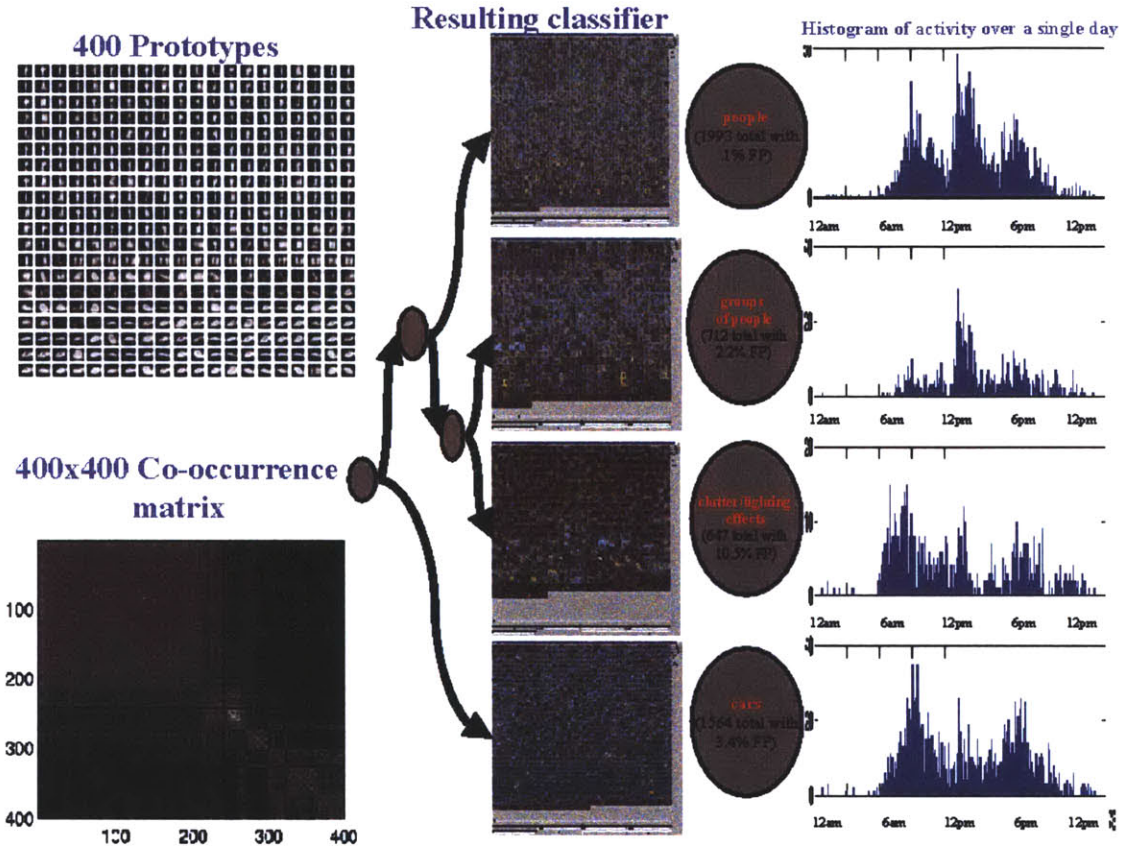


Figure 4-19: *On the left is the 400 silhouette prototypes and the co-occurrence matrix that resulted from a day's worth of tracking sequences. In the middle is the classification hierarchy which resulted, images of all occurrences of each class, and description of the classes as well as their performance relative to those descriptions. On the right are 24 hour histograms of the occurrences of each class. See web page for more higher quality images.*

regions correspond to the shirt, pants, face, and background. But apart from general characteristics of a person (e.g., size of these regions), the presence or absence of these regions is not useful in determining which person has been detected. In contrast, when trying to describe a particular person in a low-resolution color image, one would probably describe them with respect to color, e.g., "The white person with purple hair wearing a white t-shirt, blue jeans, and tennis shoes."

We are interested in modeling such images of classes of objects that are characterized by similarities and differences between image pixels rather than by the values of those pixels. For instance, images of pedestrians (at a certain scale and pose) can be characterized by a few *regions of regularity* that have fixed properties such as constant color or constant texture within the region, but tend to be different from each other. We shall refer to sets of images that fit this general description as *images characterized by regions of regularity*, or ICRORs.

This section discusses a new representation based on Multiple Observation Learning (MOL) that models the pairwise similarity between all pixels in an image patch– Similarity Templates (STs). STs can be used for detection of a class of objects, because it is invariant to the colors of particular regions. This capability can be used to align sets of images of similar objects. Further, this representation facilitates decomposition of the class of images into component regions over which robust statistics of color can be estimated. These regions can provide a compact factored description of a class of objects and facilitate recognition as well as refine detection results. Also, as shown in the following subsection, the factored representation makes occurrence-based data mining applications more feasible. The generality of similarity templates makes them an attractive representation for an attention bootstrapping system.

4.6.1 Related work

Object detection refers to detecting an instance of a particular class of object. Some examples of detection tasks are face detection [46], pedestrian detection [42], and vehicle detection [42]. Edge templates are often used for class distinctions because of their invariance to scene lighting and object color. They have similar properties to similarity templates (STs), but they are based on a measure of local differences as opposed to global similarities. The Hausdorff and Chamfer distances are mechanisms for efficiently comparing edge templates with some robustness to slight misalignments [22].

Principal Component Analysis, Multi-scale Gabor filters, and Haar wavelet functions are examples of projections of images into a lower dimensional space to facilitate recognition. Generally the coefficients in these spaces show invariance to noise within regions. Unfortunately, using these to make a general detection mechanism usually involves a complex supervised training algorithm [42], which is often run on only gray-scale images. While neglecting color information entirely is arguably ill advised, many researchers have found that learning on a color image space requires much more complexity in the classifier and extremely large data sets to train.

Recent work has shown impressive transform-invariant modeling and clustering for sets of images of objects with similar appearance. We seek to expand these

capabilities to sets of images of an object class that show considerable variation across individual instances (e.g. pedestrian images) using a representation based on pixel-wise similarities, *similarity templates.*

Jojic and Frey [28] and Miller et al. [38] have investigated transform-invariant modeling and clustering for images of a particular object (e.g., an individual's face). Their method can simultaneously converge on a model and align the data to that model. This method has shown positive results for many types of objects that are effectively modeled by a Gaussian or a mixture of Gaussians. Their work with transformed component analysis (TCA) shows promise for handling considerable variation within the images resulting from lighting or slight misalignments. However, because these models rely on an image set with a fixed mean or mixture of means, they are not directly applicable to ICRORs.

We would also like to address transform-invariant modeling, but use a model which is invariant to the particular color of component regions. One simple way to achieve this is to use edge templates to model *local differences* in image color. In contrast, we have chosen to model *global similarities* in color using a similarity template (ST).

An alternative approach to recognizing an object or class of objects is segmenting an image into color regions, representing the regions as nodes in a graph, and using graph comparison algorithms [47]. This is potentially a more general framework than our system. In our case, we are assuming that the training images are in rough correspondence as a result of the tracking algorithm. This assumption allows us to aggregate the similarity statistics across a set of images. Our method does not require discrete segmentation of each image, avoids the need for a segmentation threshold, doesn't require a complex graph structure, and has the correspondence problem implicitly solved.

While representations of pixel similarity have previously been exploited for segmentation of single images [52, 2], we have chosen to use them for aggregate modeling of image sets. Similarity templates enable alignment of image sets and decomposition of images into class-specific pixel regions. We note also that registration of two ICRORs can be accomplished by minimizing the mutual information between corresponding pixels [59]. But, there is no obvious way of extending this method to large sets of images without a combinatorial explosion.

Subsection 4.6.2 briefly introduces similarity templates in the context of multiple observation learning (MOL) discussed in the previous chapter. Then, the method of computing an "ideal" similarity template from perfectly segmented images is covered. Subsections 4.6.3 covers the mechanics of computing similarity templates without an ideal segmentation. In Subsection 4.6.4, we cover how to compare and introduce a method for bootstrapping a static attention mechanism from our motion-based detection mechanism. Subsection 4.6.6 discusses refining the alignment of the set of images. Subsection 4.6.7 covers their application to decomposing a class-specific set of images into component regions. Future avenues of research and conclusions are discussed Subsection 4.6.11.

4.6.2 The "ideal" similarity template

A similarity template is functionally equivalent to an object observation co-occurrence matrix. The primary difference is that rather than representing statistics of equivalent observations, it represents statistics of equivalent pixels, or pixels that represent the same region. As we will see later, the same type of decomposition can be used to find a probabilistic segmentation of the pixels into regions of regularity.

A similarity template S for an N-pixel image is an NxN matrix. The element $S_{i,j}$ represents the probability that pixel locations p_i and p_j would result from choosing a region and drawing (iid) two samples (pixel locations) from it. More formally,

$$S_{i,j} = \sum_r p(r)p(p_i|r)p(p_j|r), \tag{4.51}$$

where $p(r)$ is the probability of choosing region r and $p(p_i|r)$ is the probability of choosing pixel location p_i from region r.

Consider sampling pixel pairs as described above from an N-pixel image of a particular object (e.g., a pedestrian) segmented by an oracle into disjoint regions (e.g., shirt, pants, head, feet, background). Assuming each region is equally likely to be sampled and that the pixels in the region are selected with uniform probability, then

$$S_{i,j} = \begin{cases} (\frac{1}{R})(\frac{1}{S_r})^2 & \text{if } r_i = r_j \\ 0 & \text{otherwise,} \end{cases} \tag{4.52}$$

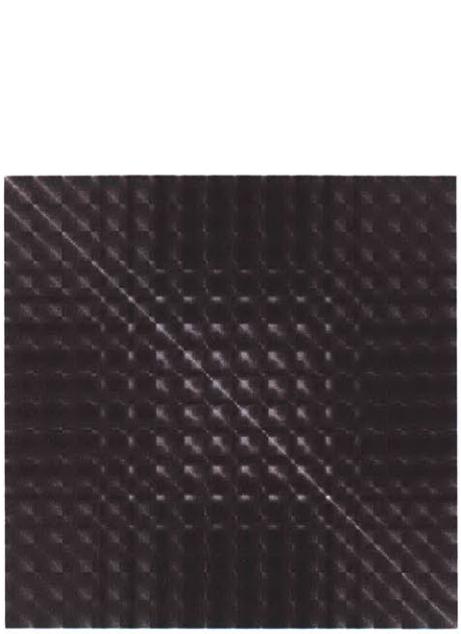
where R is the number of regions, S_r is the number of pixels in region r, and r_i is the region label of p_i. If two pixels are from the same region, the corresponding value is the product of the probability $\frac{1}{R}$ of choosing a particular region and the probability $(\frac{1}{S_r})^2$ of drawing that pixel pair. This can be interpreted as a block diagonal co-occurrence matrix of sampled pixel pairs.

In this ideal case, two images of different pedestrians with the same body size and shape would result in the same similarity template regardless of the colors of their clothes, since the ST is a function only of the segmentation. An ST of an image without a pedestrian would exhibit different statistics. Note that even the ST of an image of a blank wall (segmented as a single region) would be different because pixels that are in different regions under the ideal pedestrian ST would be in the same region.

Unfortunately, images do not typically come with labeled regions, and so computation of a similarity template is impossible. However, we take advantage of the observation that properties within a region, such as color, are often approximately constant. Using this observation, we can approximate true similarity templates from unsegmented images.

4.6.3 Computing similarity templates

Our model for similarity is based solely on color. Since there is a correlation between color similarity and two pixels being in the same region, we approximate the


(a)

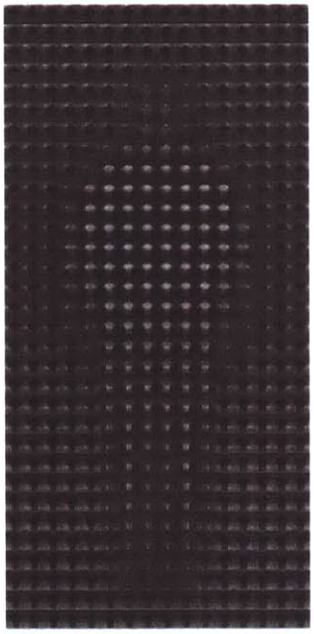

(b)

Figure 4-20: (a) The NxN aggregate similarity template for pedestrian data set. (b) An alternate view of (a). This view is a $width^2$x$height^2$ version of (a). Each sub-image represents the row of the original AST that corresponds to that pixel. Each sub-image highlights the pixels that are most similar to the pixel it represents.

corresponding value $\tilde{S}_{i,j}$ with a measure of color similarity:

$$\tilde{S}_{i,j} = \frac{1}{NZ_i} \exp\left(\frac{-||I_i - I_j||^2}{\sigma_i^2}\right), \tag{4.53}$$

where I_i and I_j are pixel color values, σ_i^2 is a parameter that adjusts the color similarity measure as a function of the pixel color distribution in the image, and Z_i is the sum of the i^{th} row. This normalization is required because large regions have a disproportionate effect on the ST estimate. The choice of σ_i^2 had little effect on the resulting ST.

If each latent region had a constant but unique color and the regions were of equal size, then as σ_i^2 approaches zero this process reconstructs the "ideal" similarity template defined in Equation 4.51. Although region colors are neither constant nor unique, this approximation has proven to work well in practice.

It is possible to add a spatial prior based on the relative pixel location to model the fact that similarities tend to local, but we will rely on the statistics of the images in our data set to determine whether (and to what extent) this is the case. Also, it may be possible to achieve better results using a more complex color model (e.g., hsv with full covariance) or broadening the measure of similarity to include other modalities (e.g., texture, motion, depth, etc.).

Figure 4-20 shows two views of the same similarity template. The first view represents each pixel's similarity to every other pixel. The columns correspond to

pixels taken in raster order. It is difficult to understand the structure in this template. The second view contains a sub-image for each pixel which highlights the pixels that are most likely to have been produced by the same region. Pixels in the shirt tend to highlight the entire shirt and the pants (to a lesser amount). Pixels in the background tend to be very dissimilar to all pixels in the foreground.

Aggregate similarity templates (AST)

We assume each estimated ST is a noisy measurement of the true underlying joint distribution. Hence we compute an aggregate similarity template (AST) as the mean $\bar{S}$ of the ST estimates over an entire class-specific set of K images:

$$\bar{S}_{i,j} = \frac{1}{K} \sum_{k=1}^{K} \tilde{S}^k_{i,j}. \tag{4.54}$$

For this quantity to be meaningful, the RORs must be in at least partial correspondence across the training set. Note that this is a less restrictive assumption than assuming edges of regions are in correspondence across an image set, since regions have greater support. Being the mean of a set of probability distributions, the AST is also a valid joint probability distribution.

4.6.4 Comparing similarity templates

To compare an estimated similarity template $\tilde{S}$ to an aggregate similarity template $\bar{S}$ we evaluate their dot product[4]:

$$s(\bar{S}, \tilde{S}) = \sum_i \sum_j \bar{S}_{i,j} \tilde{S}_{i,j}. \tag{4.55}$$

We are currently investigating other measures for comparison. By thresholding the ratio of the dot product of a particular image patch under an AST trained on pedestrian image patches versus an AST trained on random image patches, we can determine whether a person is present in the image.

4.6.5 Automatically trained static pedestrian detection

Using our tracking algorithm [55] in our laboratory environment, 32x32 patches centered on the centroid of walking pedestrians and scaled to include the entire person were automatically extracted from a live video source. The background images were extracted from the scenes randomly at approximately the same scale. Examples of these images are shown in Figure 4-21.

We estimated a single template for the background and a single template for the pedestrian class. Figure 4-22 shows the aggregate similarity template for the

[4]In our experimentation KL-divergence, typically used to compare estimates of distributions, proved less robust.

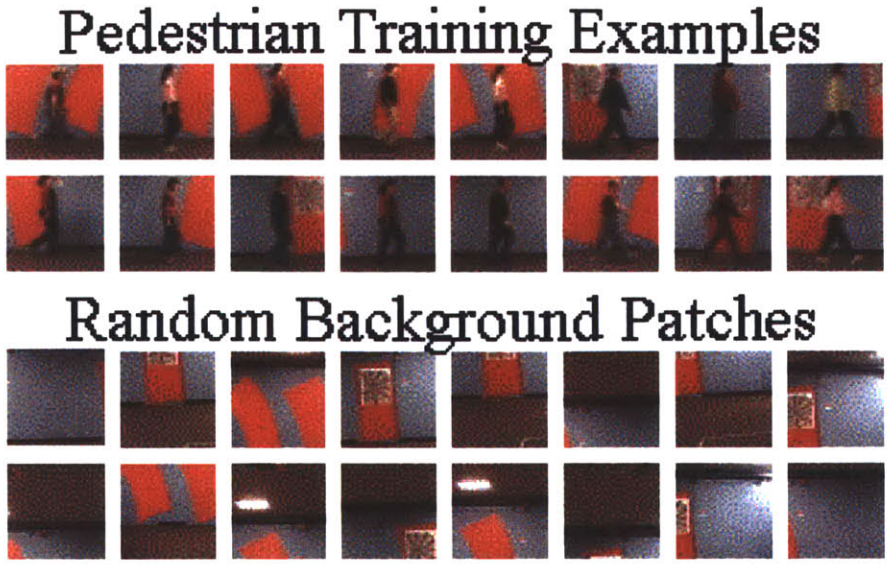
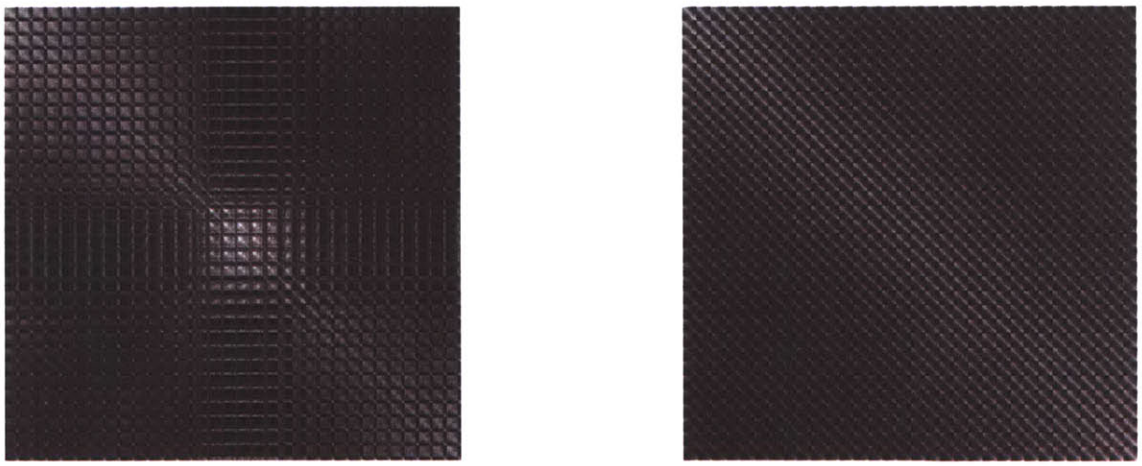
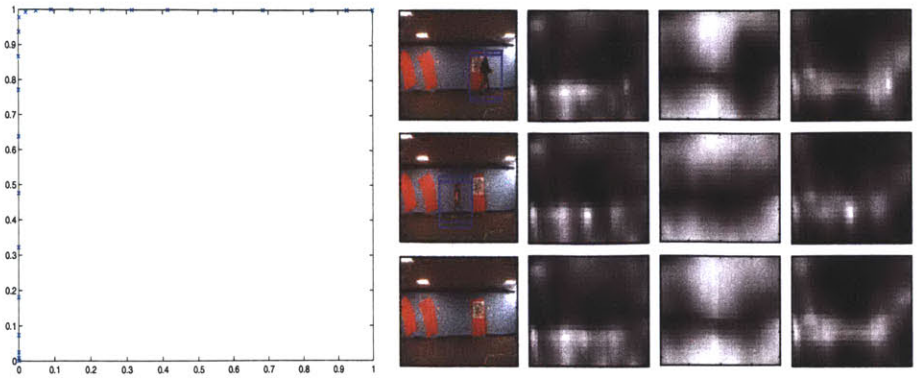


Figure 4-21: This figure shows examples of positive and negative training examples automatically extracted from the tracking system in our laboratory environment.



(a) (b)

Figure 4-22: This is the aggregate similarity template for the pedestrian data set (a) and for the background data set (b). Pixels are unrolled into a single vector such that every 32 rows represents the similarities of a single column of the image and the 32 sets of 32 rows represents the entire image's similarity template.



(a) (b)

Figure 4-23: This figure shows the ROC-curve for our detection experiments (a) with an equal error rate of less that 2%. (b) shows locations of detections for three images (two of which contain people). To the right of the images are three attention maps corresponding to: the pedestrian likelihood, the background likelihood, and the resulting likelihood ratio. The lowest values are black and the highest values are white in each image.

pedestrian training data set and the background training data set. It is difficult to infer anything about the structure of the aggregate ST without decomposing it as discussed in Subsection 4.6.7, but it is possible to observe that the pedestrian ST has a different structure than the background ST. To those who are more experienced at observing pairwise image statistics, it may be evident that the background ST shows little more than that pixels near each other tend to be similar.

A grid of overlapping image patches at the approximate scale of an average pedestrian was evaluated based on the ratio of the score (outlined in the previous subsection) of the current patch on the pedestrian AST versus the score of the current patch on the background AST.

$$rs = s(S_I, S_{ped})/s(S_I, S_{back}). \tag{4.56}$$

When this ratio exceeds a threshold T, a potential pedestrian has been detected. Figure 4-23 shows an ROC-curve and plots of the likelihood scores for different scenes. It also shows the locations of the positive detections at the threshold determined by the equal error rate(EER) of less than 2%. The EER occurs near the threshold of T that corresponds to the plane where the angle between the sample and the two models is the same.

Though one might expect even better results for this limited testing domain, we were pleased by the relative robustness to limb position. Also, detection by density approximation is difficult and should not be compared to iterative discriminant classifiers (e.g. SVMs). The reason we have not pursued discriminant methods in this paper is that they show less promise for multi-class classification and usually require extensive retraining after any change in the domain (e.g., adding new positive examples, moving to a new domain with a different type of background, adding a new class

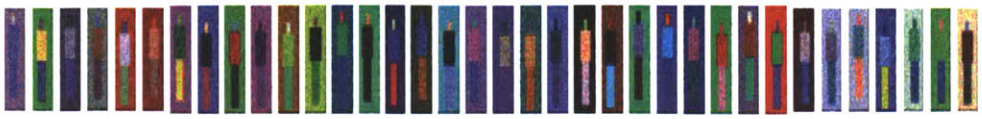


Figure 4-24: A set of randomly generated "pedestrian" images used in alignment experiments.

of object to be detected, etc.). Also, it is unclear how to extract the conditional color model (discussed below) from a discriminant classifier.

By using an iterative learning algorithm which adapted the template in a supervised fashion to learn a discriminant function, we could achieve much better results. Also, it is possible to use the conditional color model to filter some false positives because the color components are extremely unlikely for pedestrians (e.g., people wearing orange shirts and orange pants).

The comparison takes approximately one second for each image patch on a Pentium III 500MHz Processor on uncompiled Matlab code. While significant improvements in speed would result from optimized code on better hardware, it is obvious that this method of detection will have limited usefulness for certain applications without hardware implementation or hierarchical implementations. We will discuss optimizations in Subsection 4.6.10.

4.6.6 Data set alignment

Given the ability to detect objects using a Similarity Template, it is possible to investigate a more difficult problem: alignment of a set of images. To explore this problem, we created a set of 128x64 images of simulated pedestrians. These pedestrians were generated by creating four independently-colored regions corresponding to shirts, pants, head, and background. Each region was given a random color. The RGB components were chosen from a uniform distribution $[0, 1]$. Then, independent Gaussian noise was added to each pixel ($\sigma = .1$). Finally the images were translated uniformly up to 25% of the size of the object. Figure 4-24 shows examples of these images.

Using the *congealing* procedure of Miller et al. [38], we iteratively estimated the latent variables (translations) that maximized the probability of the image STs to the AST and re-estimated the AST. We were able to align the images to within .5 pixels on average.

4.6.7 The Conditional Color Model (CCM): Decomposing the similarity template

This section explains how to derive a factorized representation from the AST that will be useful for recognition of particular instances of a class and for further refinement of detection. This representation is also useful in approximating the template to avoid the $O(N^2)$ storage requirements.

An AST represents the similarity of pixels within an image across an entire class-specific data set. Pairwise statistics have been used for segmentation previously [52]. As in the previous part of this chapter, our work centers on factoring joint distributions as in [44, 19, 33, 53]. Rather than estimating two sets of marginals (conditioned on a latent class) that explain co-occurrence over different types of observations (e.g. words and documents), we seek a single set of marginals conditioned on a latent variable (the ROR) that explain our co-occurrence data (pixel position pairs). Hence, it is a density factorization in which the two conditional factors are identical (Equation 4.51). We refer to this as *symmetric factorization* of a joint density.

Also, rather than treating pixel brightness (darkness, redness, blueness, or hue) as a value to be reconstructed in the decomposition, we chose to represent pixel similarity. In contrast to simply treating images as additive mixtures of basis functions [33], our decomposition will get the same results on a database of images of digits written in black on white paper or in white on a black board and color images introduce no difficulties for our method.

As discussed in the previous example, we would like to estimate the factors from Equation 4.51 that best reconstruct our measured AST, $\bar{S}$. Let $\hat{S}$ be the estimate of $\bar{S}$ constructed from these factors. Given the number of regions R, it is possible to estimate the priors for each region $p(r)$ and the probability of each region producing each pixel $p(p_i|r)$. The error function we minimize is the KL-divergence between the empirically measured $\bar{S}$ and our parameterized estimate $\hat{S}$,

$$E = \sum_i \sum_j \bar{S}_{i,j} \log \left(\frac{\bar{S}_{i,j}}{\hat{S}_{i,j}} \right) \tag{4.57}$$

as in [19]. Because our model $\bar{S}$ is symmetric, this case can be updated with only two rules:

$$p'(p_i|r) \propto p(p_i|r) \sum_{p_j} p(r) p(p_j|r) \frac{\hat{S}(p_i, p_j)}{\bar{S}(p_i, p_j)}, \text{ and} \tag{4.58}$$

$$p'(r) \propto p(r) \sum_{p_i} \sum_{p_j} p(p_j|r) p(p_i|r) \frac{\hat{S}(p_i, p_j)}{\bar{S}(p_i, p_j)}. \tag{4.59}$$

The more underlying regions we allow our model, the closer our estimate will approximate the true joint distribution. These region models tend to represent parts of the object class. $p(p_i|r)$ will tend to have high probabilities for a set of pixels belonging to the same region. We take advantage of the fact that aligned pedestrian images are symmetric about the vertical axis by adding a "reflected" aggregate similarity template to the aggregate similarity template. The resulting representation provides a compact approximation of the AST ($O(RN)$ rather than $O(N^2)$).

Rather than performing a straight R-way decomposition of the AST to obtain R pixel region models, we extracted a hierarchical segmentation in the form of a binary tree. Given the initial region-conditioned marginals $p(p_i|r_0)$ and $p(p_i|r_1)$, each pixel was assigned to the region with higher likelihood. This was iteratively applied to the ASTs defined for each sub-region. Region priors were set to 0.5 and not adapted in

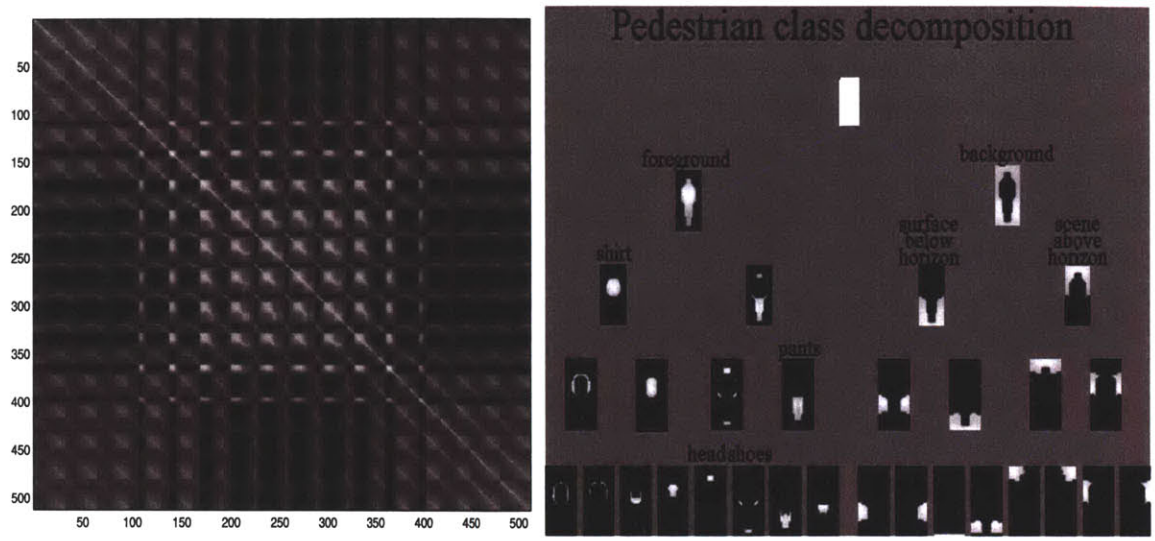


Figure 4-25: The similarity template and the corresponding automatically generated binary decomposition of the images in the pedestrian data set. The root node represents every pixel in the image. The first branch splits foreground vs. background pixels. Other nodes correspond to shirt, legs, head, and background regions.

order to encourage a balanced cut.

The probabilistic segmentation can be employed to accumulate robust estimates of statistics of the region. For instance, the mean pixel value can be calculated as a weighted mean where the pixels are weighted by $p(p_i|r)$.

4.6.8 Decomposing pedestrian images

Because the data collected at our lab showed limited variability in lighting, background composition, and clothing, we used the MIT CBCL pedestrian data set which contains images of 924 unique, roughly aligned pedestrians in a wide variety of environments to estimate the AST. Figure 4-25 shows the resulting hierarchical segmentation for the pedestrian AST. Since this intuitive representation was derived automatically with absolutely no knowledge about pedestrians, we hope other classes of objects can be similarly decomposed into RORs.

In our experience, a color histogram of all the pixels within a pedestrian is not useful for recognition and was almost useless for data mining applications. Here we propose a class-conditional color model. It determines a color model over each region that our algorithm has determined contains similar color information within this class of objects. This allows us to obtain robust estimates of color in the regions of regularity. Further, as a result of our probabilistic segmentation, the values of $p(p_i|r)$ indicate which pixels are *most* regular in a region which enables us to weight the contribution of each pixel to the color model.

For the case of pedestrian-conditional color models, the regions roughly correspond to shirt color, pant color, feet color, head color, and some background color regions. The colors in a region *of a single image* can be modeled by color histograms,

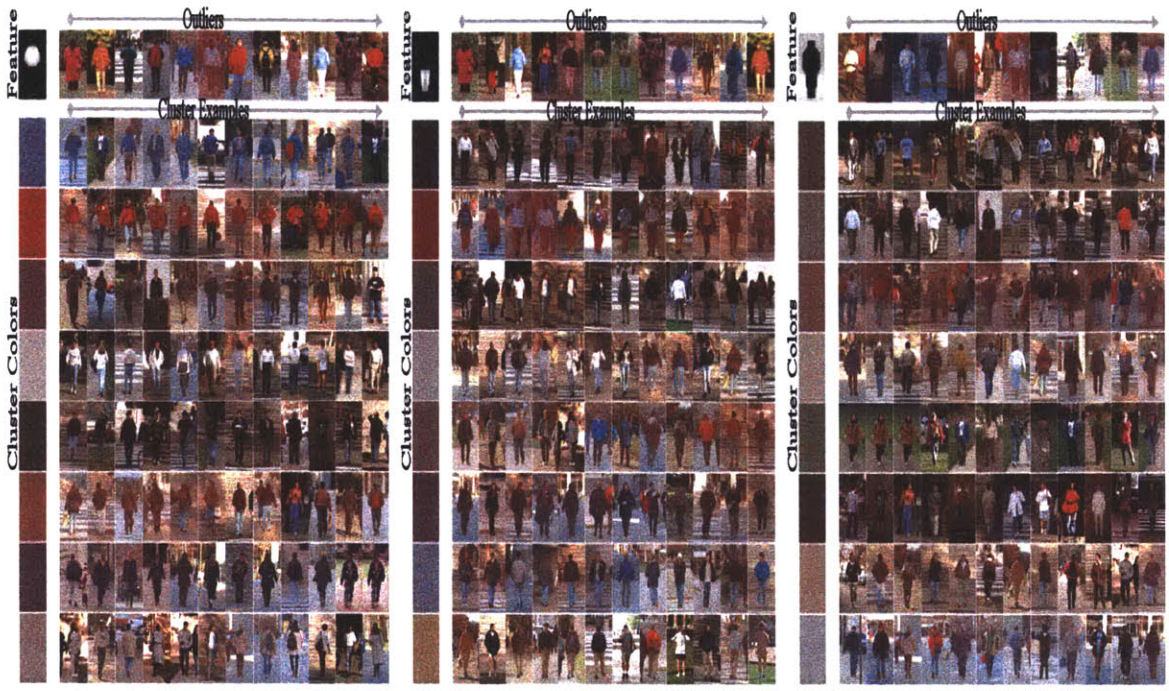


Figure 4-26: Results of automatic clustering on three components: shirt, pants, and the background. Each shows the feature, the most unusual examples of that region, followed by the 12 most likely examples for the eight prototypical colors of that region.

Gaussians, or mixtures of Gaussians. These region models can be clustered *across images* to determine a density of shirt colors, pant colors, and other region colors within a particular environment. This enables not only an efficient factored color component codebook, but anomaly detection based on particular regions and higher order models of co-occurrences between particular types of regions. For instance on a military base, military personnel's clothing may be determined by a number of factors including the weather, their sex, their rank, and the activity they are intending to perform.

Application to image indexing

To illustrate the effectiveness of our representation we chose the simplest model for the colors in each region–a single Gaussian in RGB space. The mean and variance of each Gaussian was computed by weighting the pixels represented by the corresponding node by $p(p_i|r)$. This biases the estimate towards the "most similar" pixels in the region (e.g., the center of the shirt or the center of the legs). This allows us to concisely represent the colors of each pedestrian image with 31 means and variances corresponding to the $(2^h - 1)$ nodes in a tree of height h.

We investigated unsupervised clustering on components of the conditional color model. We fit a mixture of eight Gaussians to the 924 color means for each region. Figure 4-26 shows the 12 pedestrians with the highest probability under each of the eight models and the 12 most unusual pedestrians with respect to that region for

three of the nodes of the tree: shirt color, pant color, and color of the background. Red, white, blue, and black shirts represent a significant portion of the database. Blue jeans are also very common in the Boston area (where the CBCL database was collected). Indoor scenes tended to be very dark, and cement is much more common than grass. Note: stating that fuchsia, orange, and aqua jackets are statistical outliers is not meant as a comment on any individual's taste.

Next, we tried some retrieval experiments to verify the effectiveness of the conditional color model. Figure 4-27 shows the responses to some simple manual queries. The user specified an image and a component of the model. Since the conditional color model can be precomputed, these queries are performed quickly by a single evaluation of one mean $\{R, G, B\}$ value per image. The ROC-curves show that even these simple queries correspond surprisingly well to very intuitive descriptions of the images.

4.6.9 Comparison to PCA decomposition

In comparison, a viable alternative to this approach is to use principal components analysis (PCA) to create a discriminant space. Within the most significant eigenvectors, pixels from the same region will tend to have similar projections. Also, a person wearing the same clothes would result in similar coefficients as other instances of the same person.

Unfortunately, there is no obvious way to factor this representation, which makes it difficult to represent concepts like khaki pants, t-shirt and jeans day, standing on grass, or Caucasian. Figure 4-28 shows the first 20 eigenvectors of the images in the data set. Gray-scale images were used because it is difficult to display and interpret color eigenvectors. The "shirt" is represented in all of the first few eigenvectors either by its presence or absence (negative coefficients). Beyond the first few eigenvectors, differences in lighting and correspondence of body parts play a dominant role. Also, this representation does not enable describing anomalies without complex density approximation.

4.6.10 Applicability and Future Work

While this representation shows promise, it is not ideal for every problem. First, it is expensive in both memory and computation. For detection, using a representation larger $32x32$ pixels can be tedious with today's hardware. It is possible to represent a set of sparse pairwise relationships rather than a full NxN matrix and parallel hardware would greatly improve this representation. Also, we have discussed above a way of reducing the memory requirements by factoring the joint into a set of marginal distributions. Multi-scale templates could potentially reduce the computational cost of detection experiments. On the other hand, while deriving the conditional color model is computationally intensive, it only has to be performed once. Once computed, feature extraction and queries are extremely fast.

A second restriction on the use of similarity templates is that regions must be in correspondence across a data set. This motivated our use of automatically extracted

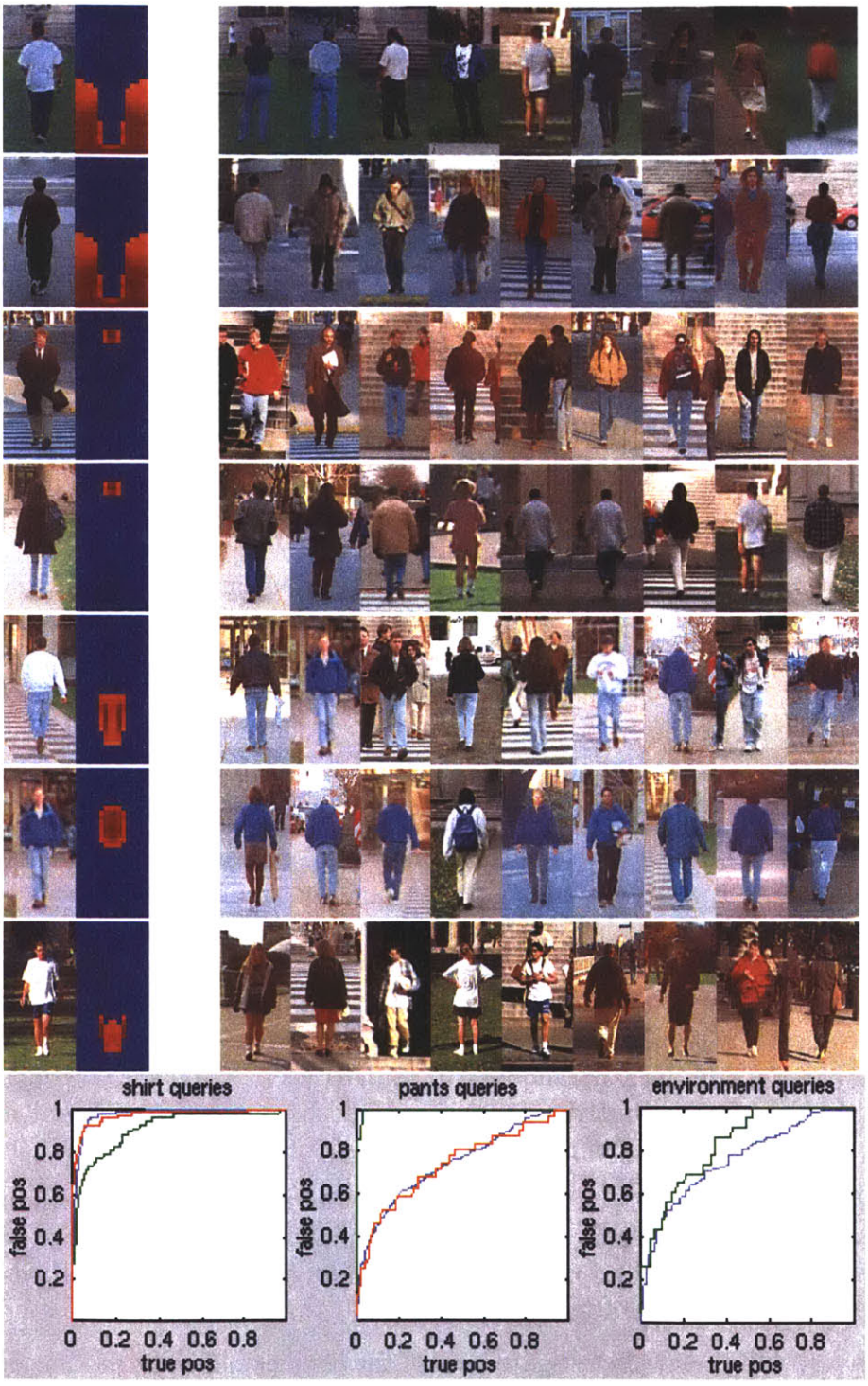


Figure 4-27: This figure shows the results of example manual queries. On the left of each column are the image and the component that were used for the query. On the right are the first 9 responses to that query. From top to bottom, the queries can be described as: standing on grass; standing on cement; faces; non-faces/dark hair; light blue jeans; blue shirts; and shorts or khakis. ROC-curves illustrate that these simple queries effectively capture intuitive descriptions of the images: red, blue and white shirts; red pants, dark pants, blue jeans; and cement/pavement and grass.

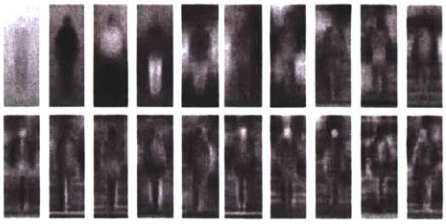


Figure 4-28: This figure shows the first 20 eigenvectors of gray-scale images in the data set.

pedestrians using motion segmentation to enable static detection. Our detection mechanism can be used to refine correspondence, but this does not deal with the variability within the class of objects.

We plan to investigate hierarchical templates that break down a large training set into sets of templates that have regions in relative correspondence. For instance, three or four templates should be capable of dealing with the variability in limb position within a pedestrian's walking cycle. More templates would be needed to handle variability in viewing angle. This could improve both detection and extraction of factored models, but is beyond the scope of this paper. We are investigating representing other classes of objects, such as vehicles and faces, but the inherent variability in shape and configuration emphasizes the necessity of developing hierarchal templates.

A third restriction is that we are only using a simple measure of pairwise similarity-color similarity. A completely uniform shirt exhibits similarity between all the shirt similar to the aggregate model. A black and white checkered shirt exhibits similarity between the black shirt pixels and similarity between the white shirt pixels, which is still somewhat similar to the aggregate model. But a "Hawaiian" shirt or camouflage shirt will exhibit few regularities in the shirt region. In that case, detection would rely on regularities outside the shirt and lack of regularities between the surroundings and the shirt. In the future, similarity templates could be applied to different modalities including texture similarity, depth similarity, or motion similarity.

In order to compete with current object detection mechanisms, we will investigate applying discriminant methods to similarity templates. While it promises to be a bit cumbersome, we believe it has the potential to show good generalization. It may also result in a more understandable representation.

Finally, we will further investigate uses of the conditional color model. First, each region can be represented by a more complex model like a mixture of Gaussians or a color histogram. Second, relative values between regions may be useful to determine features like "is wearing backpack" or "is wearing shorts".

The clustered color model also has application to learning grounded labeling. Given a set of images of an object type and a set of text or audio descriptions of those images, the clustered color model will facilitate binding words to concepts as was done with toys of a single color as in [49].

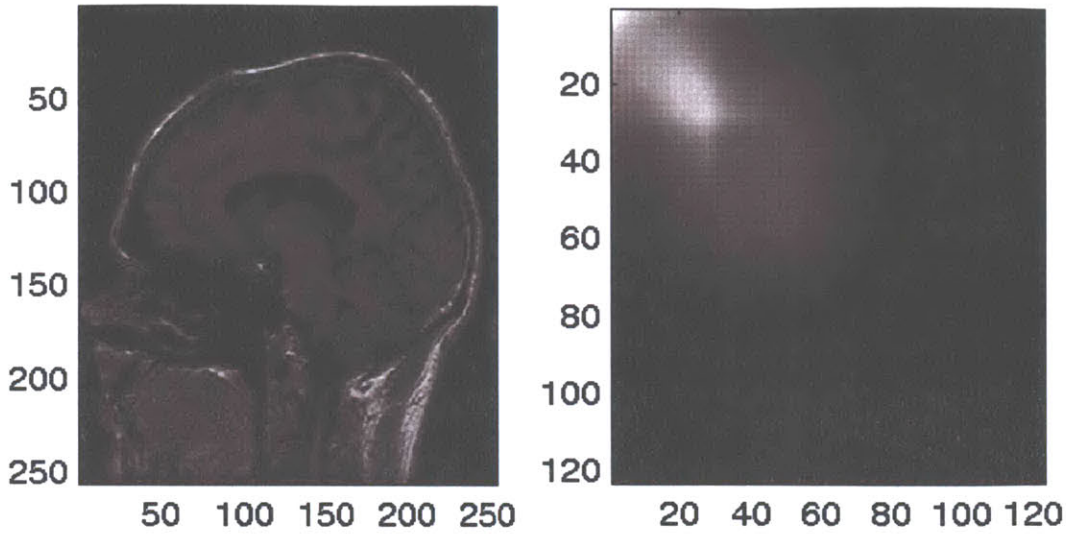


Figure 4-29: This figure shows an magnetic resonance image (MRI) (a) and the corresponding pixel-value co-occurrence matrix measured from Gaussian weighted image patches (b).

4.6.11 Similarity Template conclusions

While this representation shows promise, it is not ideal for many problems. First, it is expensive in both memory and computation. Here, we are only using a simple measure of pairwise similarity–color similarity. In the future, similarity templates could be applied to different modalities including texture similarity, depth similarity, and motion similarity.

While computationally intensive, we believe that similarity templates can provide a unified approach to the extraction of possible class-specific targets from an image database, alignment of the candidate images, and precomputation of meaningful features of that class. For the case of pedestrians, it could detect potential pedestrians in a database, align them, derive a model of pedestrians, and extract the parameters for each pedestrian. Once the features are computed, query and retrieval can be done efficiently.

We have introduced a new image representation based on pixel-wise similarity. We have shown its application in both alignment and decomposition of pedestrian images.

4.7 Co-occurrence based tissue class modeling

The final example of Multiple Observation Learning (MOL) is a system which uses spatially local co-occurrence of pixel values to determine a tissue class model for magnetic resonance images (MRIs). This final example of MOL shows the generality of this approach and the promise of this approach for both high-level modeling in

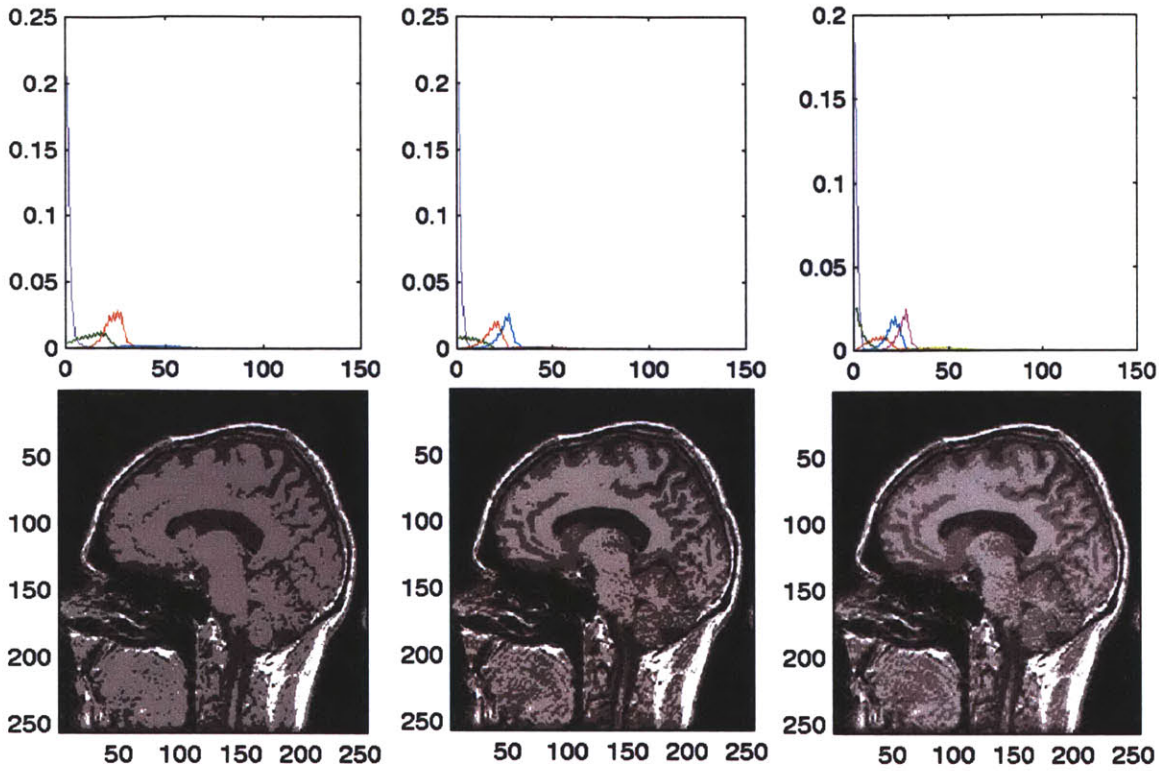


Figure 4-30: This figure shows four (a), five (b), and six (c) latent tissue class models.

low-level intermediate modeling.

In an ideal situation, an overcomplete segmentation of the MRI could be used to estimate the probability of pixel-values pairs being drawn from a random segment of the MRI. Pixel value pairs likely to be produced by the same tissue type would have higher co-occurrence. Pixel value pairs that were not likely to be produced by any tissue type would have a low co-occurrence.

Assuming the pixel values in these segments were of one tissue type and exhibited the expected variation in that tissue type, latent class models estimated from the measured co-occurrence should correspond to the tissue type models.

Since a tissue-based segmentation is not available, we make the assumption that the tissue type in a random, local window tends to be uniform. Patches near edges of tissue regions will contain observations from multiple tissue types, but our assumption is that these will be statistically insignificant. Figure 4-29 shows an example MRI image and the co-occurrence of pixel values (1 to 128) in Gaussian weighted image patches ($\sigma = 5$). Because of the large background, dark pixels ([1,7]) have the highest co-occurrence. The other major peak is in the range of $[18, 30]$ which corresponds to the regions of gray/white matter.

The occurrence values are best fit by two Gaussians. More than two Gaussians produces unreliable tissue models. Figure 4-30 shows the latent tissue class models derived from the observed co-occurrences. The four class model roughly corresponds to models for air, cerebral spinal fluid, white/gray matter, and skin. The addition

of the fifth model does little to alter the four class model except in splitting the white and gray matter tissue class into two classes. The addition of the sixth model introduces another background class while having little effect on the white matter and gray matter tissue models.

These results are surprising given that no segmentation was available. They show that the assumption of local uniformity in latent class is reasonable. Since the model is non-parametric, no scaling or warping of the space would greatly affect these results. In fact, grayscale pixel values could be replaced by color values or texture values. This is an area of future work.

Chapter 5

Meta Modeling

The previous chapters have illustrated how to exploit many different types of structure in an environment without requiring knowledge about the particular environment to be specified. We have shown that we can characterize the type of objects, the activities of the objects, the characteristics of the appearance of particular types of objects, and other aspects of the environment. This chapter covers automatically modeling temporal structure as well as characterizing objects and activities that are not typical of an environment.

Section 5.1 discusses exploiting temporal context. Thus far in this document, the time an object was observed was not considered. In many environments, the temporal context of an event can be extremely important. We introduce a system which first determines temporal context cycles and then exploits the context cycle to characterize the activity of the scene. In an office setting, we determine that there are 24-hour and 7-day cycles of activities. Using this information we can characterize activities at unusual times of the day. We could also characterize days with unusual amounts of activity (e.g., vacation days and student visit weekends). In a traffic intersection, we learn the traffic light cycle. This enables detection of events like people running traffic lights. Section 5.2 introduces four types of anomaly detection: observation anomalies, co-occurrence anomalies, temporal anomalies, and anomalous activity periods.

5.1 Modeling and exploiting temporal context

The activity model discussed thus far does not incorporate temporal context. In this section we outline a system to determine and exploit context cycles that result from repetitive patterns, e.g., traffic cycles, tour schedules, 8-hour factory floor activity patterns, daily patterns of activity, weekly patterns of activity, etc. Automatically determining if and when these context cycles exist can facilitate

- **Scene classification-** Different periods may be characteristic of different classes of environment. A 75 to 95 second cycle can be indicative of a traffic intersection. A two to three minute cycle can be indicitive of a amusement ride. Strong weekly patterns of activity may be a counter-indication of a natural en-

vironment. Longer characteristic cycles may be indicitive of different weather patterns in different areas of the world.

The exact period of a traffic cycle or how that period changes over a daily cycle could be a very strong indicator of a particular intersection. This would facilitate the search for pairs of scenes which have direct visual overlap or other relationships.

- **Context-sensitive modeling-** It could be advantageous to build seperate models for different parts of context cycles. The shapes of objects seen during the day and at night can be very different (headlights vs. complete vehicle silhouettes). Other aspects of the model may also vary with the time of the cycle. Hence, a more articulated model may result from multiple models used to represent different segments of a context cycle.

  We believe that these methods for modeling and exploiting context will also be useful in modeling other types of context. For instance, weather (e.g. sunny, cloudy, rainy, etc.) is not cyclic, but it can alter the visual observations of most objects. Weather can even alter the expectation of the activities that occur in an environment. For example, people do not have shadows but carry umbrellas on rainy days. They tend to move faster or loiter in covered spaces.

- **Context-sensitive anomaly detection-** While some behavior or appearance can be characterized as unusual regardless of the context, many aspects of typicality are context sensitive.

  For instance, deliveries are typical to most office buildings, but deliveries at 2am may not be. Also, a person accelerating through a traffic light is not unusual unless it occurs during the wrong stage of the traffic light cycle. Traffic on a busy highway may be stop and go during morning rush hour but not in late evening.

- **Finding repetative activities-** Often sparse, repeated events occur at certain times in the periodic cycle of an environment, e.g., food trucks, UPS deliveries, meter checkers, weekly group meetings, etc. Finding such events would involve an extremely large search even if only unusual events were used.

Subsection 5.1.1 discusses the method we used to determine the context cycles. Subsections 5.1.2 and 5.1.3 show examples of daily/weekly patterns of activity in an office environment and a traffic light cycle in an urban environment shown in Figure 5-1(a) and (b).

5.1.1 Determining the context cycle

In this work, we are assuming that context cycles are locally periodic. A context cycle with wide variation in cycle timing or extensive drift in the cycle period would necessitate a more complex model. Our model is well-suited for many significant types of periodic activity cycles.

(a) (b)

Figure 5-1: This figure shows the two environments used as examples in the cyclic context modeling experiments. The first is an office environment in which a refrigerator was placed that contained inexpensive soft drinks. Most of the activity in this area was graduate students using the refrigerator or others passing through the environment. The second example is a traffic intersection in Boston.

First, we define the wrapped distribution of observation activity as

$$p(i|\bar{t_j}) = \frac{\sum_{o_x} \delta_{t_x,\bar{t_j}}}{\sum_{o_x} 1} \tag{5.1}$$

where o_x is an index of object observations and $\delta_{t_x,\bar{t_j}}$ is one if the time of the observation falls within the time interval defined by $\bar{t_j}$. For instance, $\bar{t_0}$ could indicate the time interval from midnight to 1am in a 24 hour time cycle. To determine if a probability cycle exists, we evaluate the entropy for an integral number of periods.

$$e = \frac{\sum_{\bar{t_j} \in \bar{T}} p(i|\bar{t_j}) \log p(i|\bar{t_j})}{e_{max}} \tag{5.2}$$

where T is a set of repeating time intervals which cover the entire period and do not overlap and e_{max} is the maximum entropy (the entropy for a uniform distribution over $\bar{t_j}$). We used a histogram with 100 equal-sized bins for our experimentation.

Because cycle drift exists even in the most regular cycles (e.g., daylight savings time), we evaluate average entropy of windows of exactly Q periods. Q must be an integer value to avoid estimation errors in entropy. If too few periods are used, the entropy estimates will be too noisy. If too many periods are used, drift will cause unreliable entropy estimates. In our experiments using three periods produced robust results.

By evaluating the times that define the minimal entropy configurations, different potential context cycles can be determined. The following two subsections show two examples for this process. In the Section 5.2, we will discuss some ways of taking advantage of the knowledge of the existence and lengths of periods for anomaly detection.

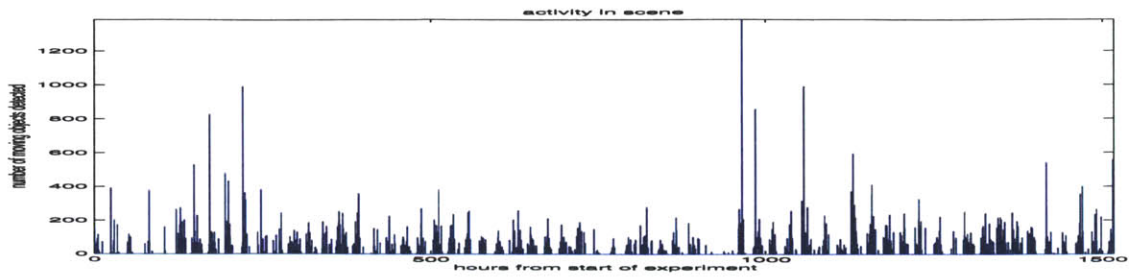


Figure 5-2: This figure shows the number of objects detected in the office environment in uniform time periods over a period of 63 days. The day and night cycles are easily visible. The weekend days are generally less active than week days. Spikes in activity result from individuals loitering in the scene for long periods of time.

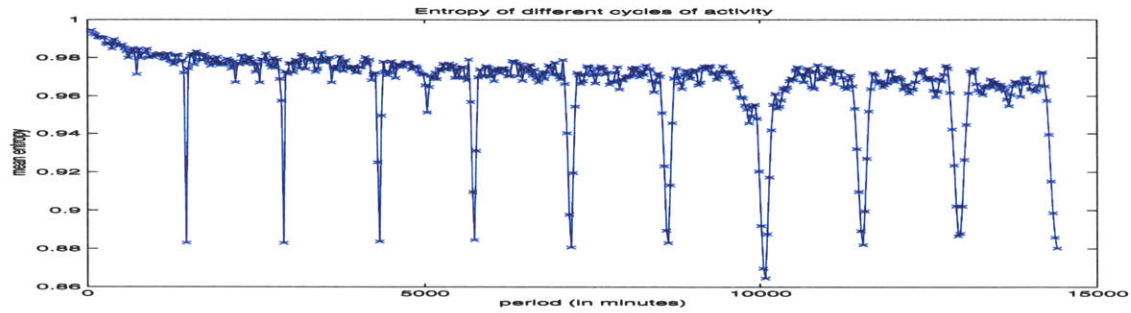


Figure 5-3: This figure shows the entropy for different activity period lengths in our office environment. The first significant drop in the entropy is at exactly 24 hours. Every multiple of 24 hours has nearly the same entropy until the 7 day period.

5.1.2 Office context cycles

Our first experiment involves a camera observing our vision laboratory which contains a water dispenser, a refrigerator stocked with soft drinks, and a recycle bin. The empty scene is shown in Figure 5-1(a).

Figure 5-2 shows a plot of the activity levels at that refrigerator for the extent of the experiment which lasted from February 13, 2002 to present (April 19, 2002). The 24 hour cycle is readily apparent. Most individuals are in the area for less than 10 seconds. Spikes of activity usually result from individuals who loiter in the area for long periods of time. Weekend days tend to exhibit less activity except in certain cases corresponding to our student visit weekend and conference deadlines.

Figure 5-3 shows the entropy for different period lengths. The minimal entropy corresponds to the 7 day cycle. The next lowest set of minimal entropy periods have a common factor in the 24 hour cycle. This can be contrasted with the spectra shown in Figure 5-4. The spectra shows a strong peak for a single day as well as smaller peaks corresponding to 3.5 days and 7 days. It is less evident that the two most significant periods could be extracted in this case and it is not obvious that the frequencies representing the largest amount of power will result in a predictive model of the amount of activity at different times in the cycle. While the spectra

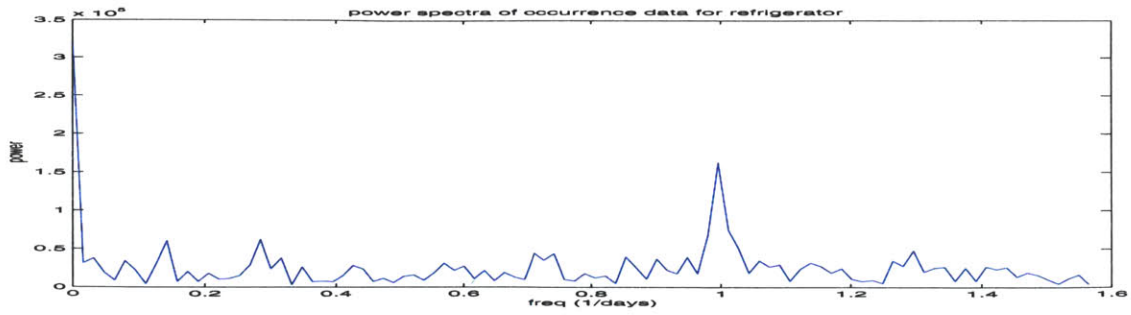


Figure 5-4: This figure shows the power spectra of the unwrapped office environment histogram. After removing the zeroth component that corresponds to the DC offset, the most significant component is the single day period. The next is the 1/2 week period. The next is the 1 week period.

shows the desirable property of not representing a cycle at multilples of a cycle, it may not capture significant power at a particular frequency if the signal is balanced at that frequency (is not correlated with a sinusoidal signal at the frequency).. For these reasons, we use the entropy measure to determine the significant periodicities in the activity histograms.

Figure 5-5 shows the histograms for the activity wrapped in a 24-hour cycle and a 7-day cycle. The average day shows that the workday at the MIT Artificial Intelligence Laboratory starts after 8am and ends for some at 5pm, for some at 7pm, and for some in the early morning. There are slight peaks in activity near lunch and a common afternoon meeting slot. There very little activity from 3am to 7am.

The average week shows less activity on Saturday and Sunday, but probably much more than in a typical office environment on those days. Friday shows heightened evening activity due to a open social event on the same floor. The weekend of the third entire week exhibited heightened activity due to the AI/LCS student visit weekend. The following two weekends exhibited even less activity than normal due to spring break.

While the 24 hour and seven day periodicity of tracking data may seem obvious, there are many situations in which there may be other significant periodicities or the periodicity may not be significant. Researchers studying the circadium rhythm of mice with genetic knock-outs are interested in how the activity cycles of these mice vary when there are no external indications of the diurnal cycle. If our system was employed to watch wildlife, the 7 day cycle would have no significance. Some businesses have significant two or three week cycles. Factories often have 8-hour work cycles. Timed-intersections are an example we will pursue in the next subsection.

5.1.3 Timed-intersection context cycles

Figure 5-1(b) shows an intersection in Cambridge, Massachusetts. Figure 5-6 shows the amount of activity in this scene for approximately one hour. Unfortunately, the amount of raw activity does not indicate a strong periodicity for this scene as shown in Figure 5-7. The minimum entropy is less than 0.35% less than the maximum entropy.

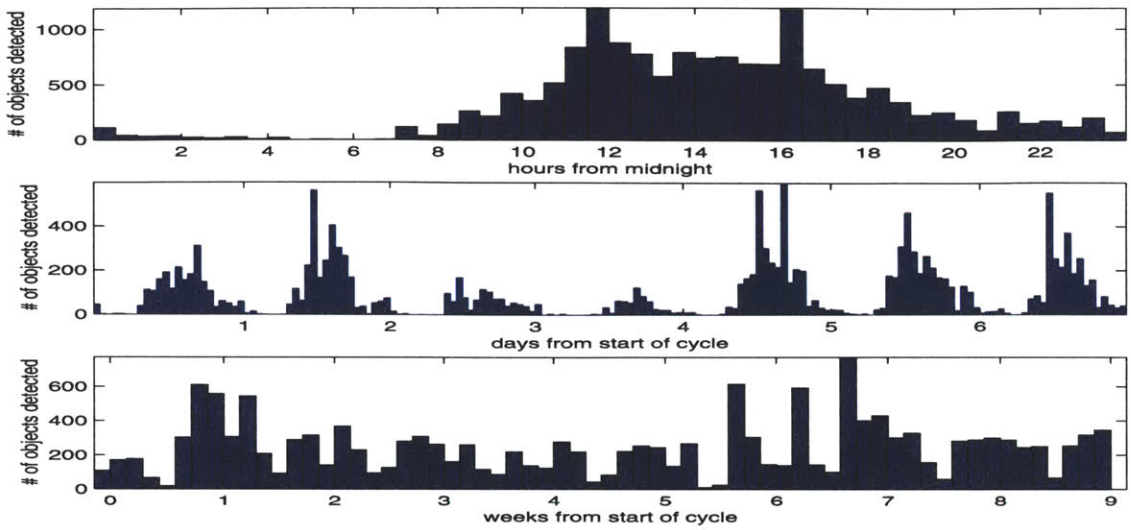


Figure 5-5: This figure shows wrapped histogram of activity for 24-hour and 7-day periods and a daily histogram for our office experiment. The amount of activity is shown in half-hour, one-hour, and one-day blocks respectively. The week period begins at midnight on Thursday morning. The entire 63 day experiment lasted from February 13, 2002 to April 19, 2002.

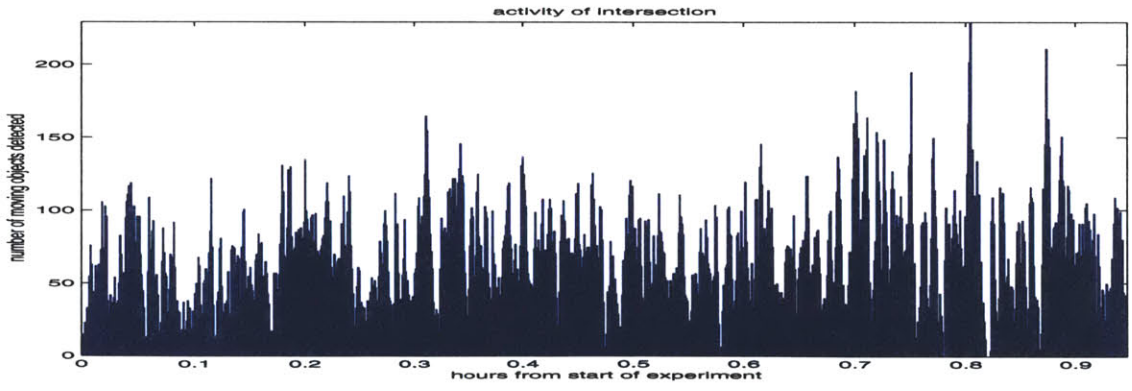


Figure 5-6: This figure shows the amount of objects that were detected in intersection environment over a period of approximately and hour. Little can be said about the periodicity using the class-independent amount of activity.

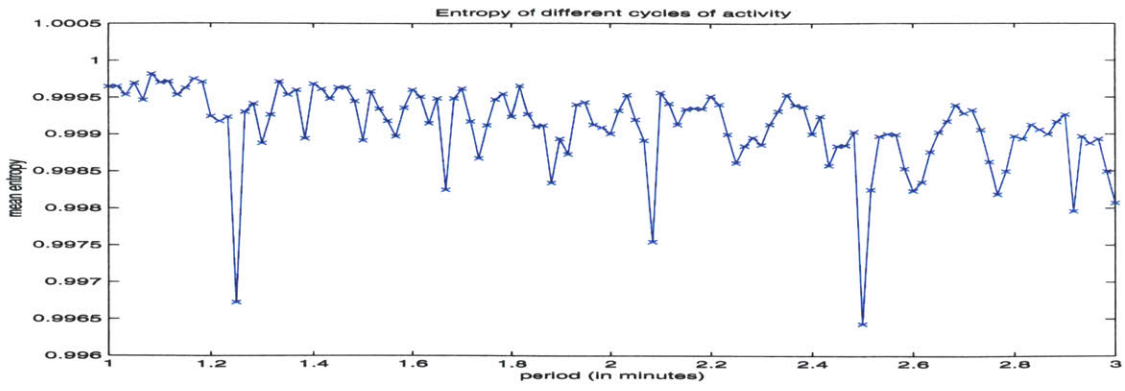


Figure 5-7: This figure shows the entropy of the gross activity in the intersection for different period lengths. Minimal entropy periods correspond to multiple of the 1.25 minute (75 second) period.

This is because the amount of class-independent activity is relatively independent of the cycle (there are many cars and pedestrians visible during the entire traffic light cycle). What defines the cycle is the type of activity that occurs during each phase.

After clustering the activities in the scene into eight clusters using a flat 8-way MOL estimation, a class-conditional entropy can be used to robustly determine that there is a 75 second activity cycle. The class conditional entropy is the expected entropy for the activity of each independent class. Figure 5-8 shows the prototype states for each type of activity as well as the observations that were in MOSs classified to each of the eight classes.

Classes 3 and 7 are primarily vehicle traffic entering from the east and exiting west. Class 1 contains primarily vehicle traffic entering from the east and exiting north. Classes 4 and 6 are primarily vehicle traffic in the opposite direction. Class 2 corresponds to a tight cluster of activity that enters from the west and exits east. Class 8 is primarily traffic exiting east from all directions. Class 5 is primarily pedestrians.

This course clustering of activity results in the activity histograms in Figure 5-9. The class conditional entropy has a minimum at periods of 1.25 minutes (75 seconds) and 2.5 minutes (150 seconds). This corresponds to one and two periods of the traffic light at this intersection during this time of the day. The entropy corresponding to a single traffic light cycle is more than 6.5% below the maximum entropy.

Figure 5-11 shows the class-conditional wrapped histograms for the 75 second cycle. Class 6 has a peak at 40 seconds which corresponds to the N-S traffic. Classes 2, 3, and 7 peak after 65 seconds corresponding to E-W traffic. This portion of the cycle is the longest. Class 5 correspond to pedestrian traffic and shows no periodicity.

5.1.4 Sub-cycle temporal analysis

Within a cycle there are periods of similar activities. We can use the MOL framework to find temporal models of similar activities. Figure 5-12 shows eight sets of 45 histograms corresponding to each activity cluster's activity in each full traffic cycle in the intersection model. The co-occurrence of similar activities in each of the 75

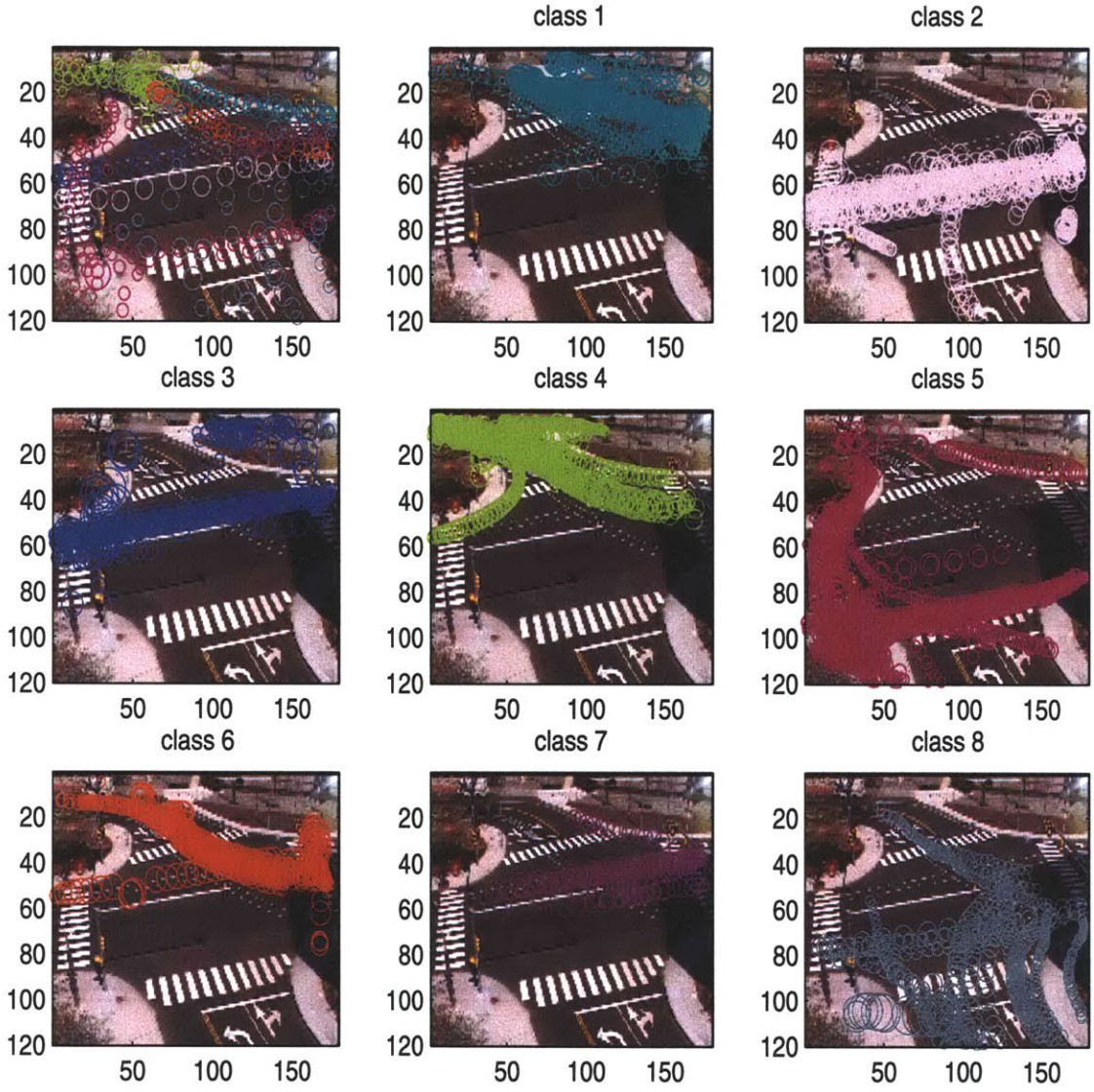


Figure 5-8: This figure shows the prototype observations and observations from MOSs for each cluster of activity. North is up east is to the right. See text for description of classes.

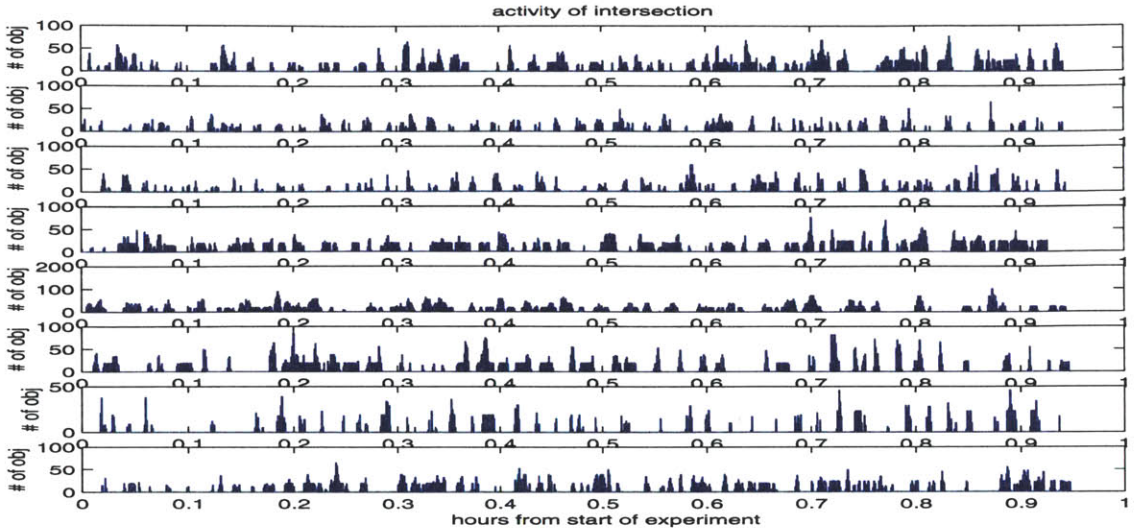


Figure 5-9: This figure shows the class-conditional histograms of activity. Many of them exhibit the periodicity of the intersection.

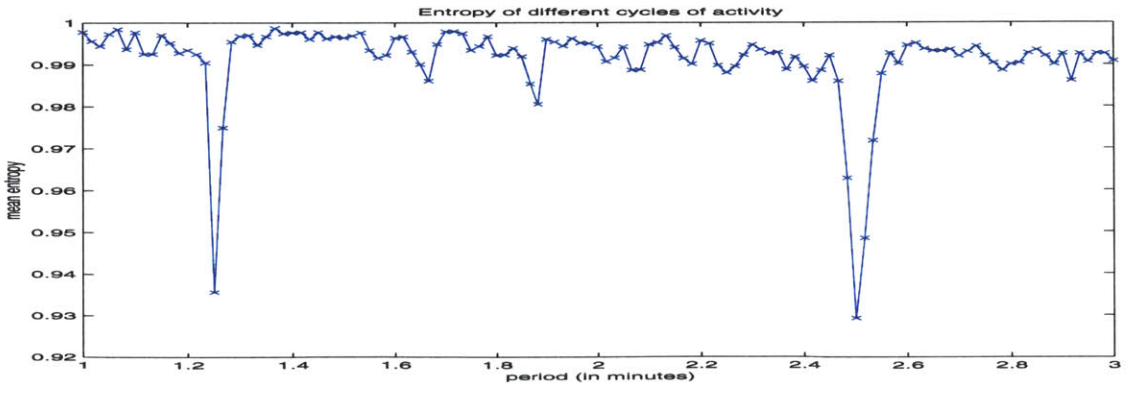


Figure 5-10: This figure shows the entropy of the class-conditional activity in the intersection for different period lengths. Minimal entropy periods correspond to multiple of the 1.25 minute (75 second) period.

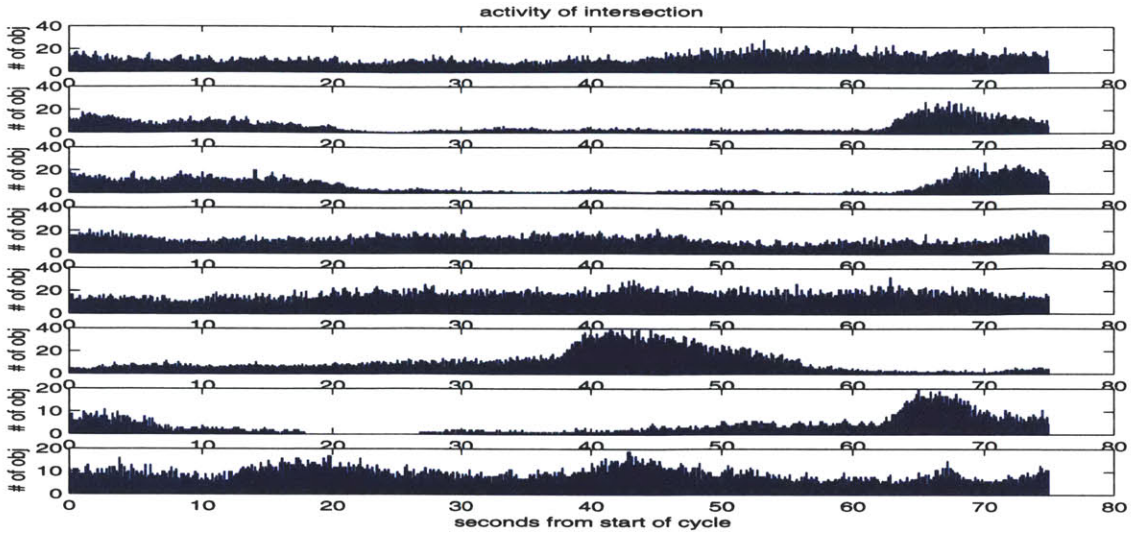


Figure 5-11: This figure shows the wrapped class-conditional histograms of activity assuming periodicity of 75 seconds. With only eight clusters of activity to describe the activity of the scene, the temporal structure is visible. As expected, classes two, three, and seven seem to operate in phase whereas class 6 is opposite.

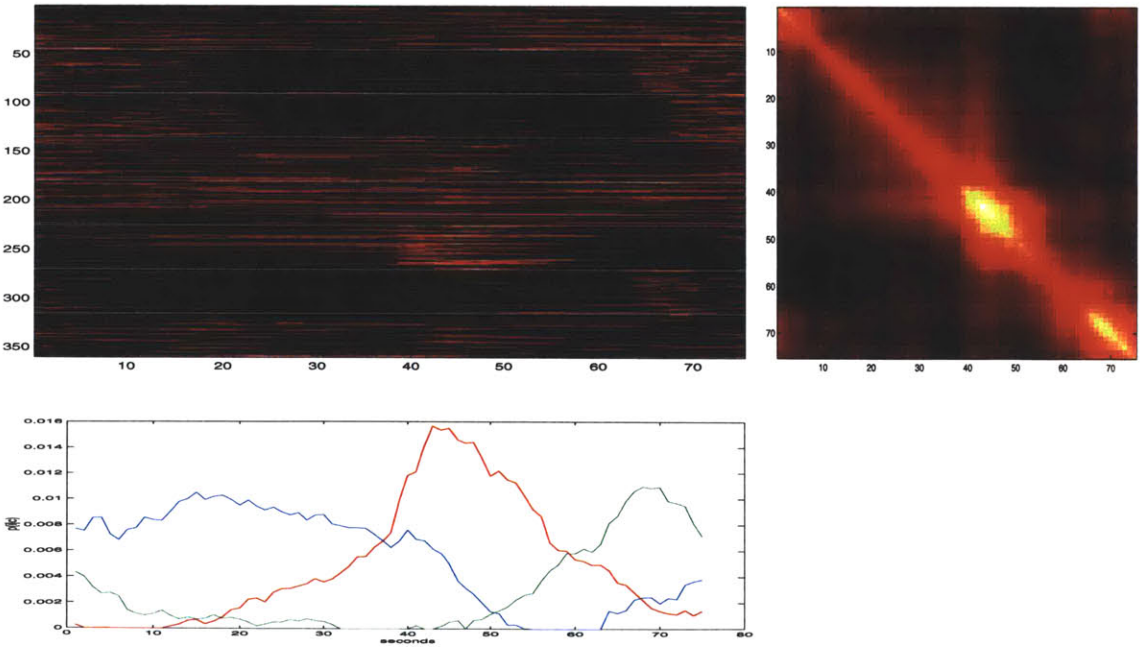


Figure 5-12: This figure shows the amount of activity for within each cycle for all eight activity clusters (a), which can be used to approximate the temporal co-occurrence matrix (b). This matrix can be used to determine a soft temporal segmentation of the traffic cycle into three classes (c).

seconds can be estimated by using using each histogram as an MOS. Figure 5-12(c) shows the resulting soft temporal segmentation. Three resulting temporal distributions represent primarily N-S events, E-W events, and the remainder of the cycle.

5.2 Anomaly detection

Given a new object observation, one can determine the most likely class labels. If a significant number of similar object observation sets have been observed, the new object observation can be characterized by the closest cluster (or its deviation from that cluster). Unfortunately, most scenes will present events which have never occurred or occur so rarely that they are not represented in the clustered activities. If the observation set is not typical for the scene, the closest cluster will not be representative of that observation set. These observation sets can only be described as anomalous. In many cases, these anomalous events are of more interest than the regular activities.

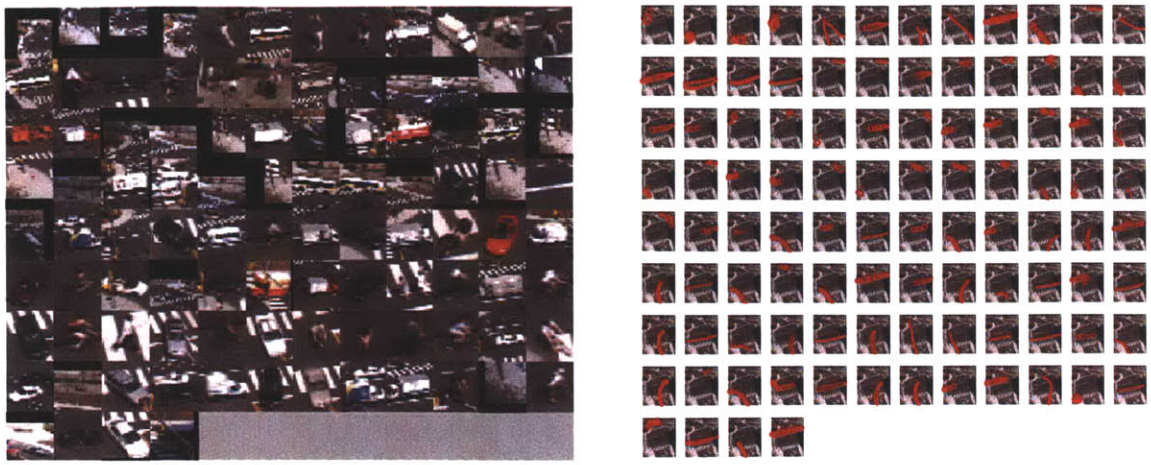
We define four types of atypical events. First, an observation anomaly occurs when an object produces observations that are atypical. Second, a co-occurrence anomaly occurs when an object produces sets of observations that are well-represented in the codebook but whose co-occurrence is atypical. Third, a temporal anomaly occurs when a characteristic activity occurs at an uncharacteristic time. Finally, anomalous activity periods correspond to periods of heightened or depressed activity levels relative to what it expected in that context. These four types of anomalies are covered below.

Not surprisingly, these anomalies correspond to the multiple parts of the probabilistic model of activity in the environment. Each aspect that has been nonparametrically, probabilistically modeled can be exploited to determine anomalous activity. The four types of anomalies correspond to outliers with respect to the codebook, co-occurrence statistics, wrapped histogram, and aggregate histogram.

Observation anomalies

The likelihood of the observations in an MOS is the geometric mean of the likelihood of the entire set of observations. It is generally easier to compute the log of this value–the expected log likelihood of the observations.

For the intersection environment the 100 least likely MOSs corresponded to 17 cars exiting south, 6 pedestrians exiting south, east-west traffic of unusual size(6 buses, 17 trucks, 3 18-wheelers, 4 combinations of vehicles, 11 bicyclists in Boston traffic), 4 groups of pedestrians in crosswalks, 6 individual pedestrians walking/biking/blading through the center of the intersection, one car nearly exiting south but turning east, and 26 tracks caused by lighting effects. The fifth and seventh anomalies were staged activities (walking through the middle of the intersection and back) which were intended to be anomalous. Four other individuals also passed through the middle of the intersection including two pedestrians, a rollerblader, and a bicyclist. An additional staged activity (walking over seedling grass) was not detected as anomalous because it was surprisingly common for that area of the scene.



(a) (b)

Figure 5-13: This figure shows example images and plots of locations and sizes of the top 100 observation anomaly sets.

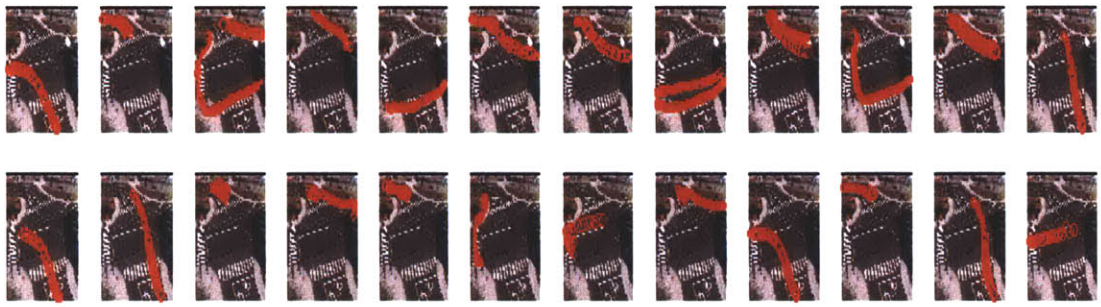


Figure 5-14: This figure shows example images and plots of locations and sizes of the top 24 co-occurrence anomaly sets.

Images and plots of activities of these are shown in figure 5-13. For the most part, these anomalies correspond to sequences in which objects were observed where they were not expected or were of an unexpected size. In some cases, these were the result of tracking failures. The first four anomalies were lighting variation that was tracked across the entire scene. Three of the next four were pedestrians moving through the middle of the intersection. An additional layer of clustering on the anomalies would allow a quick summarization of many of the major sources of anomalies.

Co-occurrence anomalies

Similarly, the likelihood of the co-occurrence of the MOS is the expected log likelihood of the observation pairs in an MOS. These anomalies tend to be object state sequences which show unusual pairs of observations, for instance, a u-turn on a road where cars don't tend to u-turn. Often these anomalies will highlight tracking failures where two objects are tracked as the same object. Figure 5-14 shows the top 20 co-occurrence anomalies at the intersection. Many of them look like regular activities since these

anomalies can result from hesitation in an unusual place or for an unusual amount of time. The third and eighth anomalies resulted from tracking failures in which multiple objects were tracked as a single object.

Temporal anomalies

The likelihood of an MOS can also be characterized by the likelihood of an MOSs of that class occurring at that time. For a 24-hour periodic environment, the amount of activity expected at a particular time is the product of the probability of seeing activity on the corresponding day and probability of seeing activity at that time of that day. An empiricle estimate of the probability of an activity at the time of a cycle is the wrapped histogram (e.g., an average day). The probability of activity on a particular day can be assumed uniform, relative to the amount of activity on that day, or estimated using a larger context cycle (e.g., a weekly context cycle). We used a weekly activity model in our experiment.

In our office environment, activity that occurred at unlikely times tended to be on weekends and early in the morning. Six of the top ten temporal anomalies occurred on weekends and eight occurred between 11pm and 4am.

Anomalous activity periods

Activity levels can also be used to characterize periods of unusual activity. By evaluating the probability of the locally agregated activity levels, the most atypical activity periods can be determined. This estimate could be conditioned on temporal context or an unconditional estimate.

We used a simple context-independent activity model in our office experiment. Periods of large amounts of activity are most unusual. Other environments may have an expected level of activity and too much or too little activity would be atypical. In the office environment, the periods of abnormal accumulated activity corresponded to individual's who loitered in the region for one reason or another. After grouping adjacent anomalous periods, the top twenty anomalous periods were primarily people servicing the equipment, people having technical conversations, and people changing the environment in some way. Table 5.1 lists descriptions of the top twenty anomalous periods.

5.2.1 Anomalies on other modalities

These general mechanisms apply to all modalities that can be clustered. For instance, object silhouette sequences can be characterized by any silhouettes that are of a very unusual shape, by objects that show variation in shape that is not expected, by an unexpected shaped object in a particular temporal context, or by aggregate numbers of objects of different classes in an area.

Num	Description of Anomalous Activity Periods
1	Six people getting water and one person loitering within three minutes.
3	Person #1 and person #2 having conversation about system.
4	Person #1 and person #3 working on the board.
5	Father and boy playing ball.
6	Person #1, person #2, and person #3 talking about system.
7	Person #2 stocking the fridge.
8	Ten people from meeting and two people in conversation research.
9	Person #2 changing the sign and stocking fridge and one conversation.
12	Person #1 stocking the fridge and three person conversation.
13	Person #2 replacing backdrop.
14	Person #2 moving cups and tea supplies to the coffeespace. and making the first cup of tea.
15	Students getting soda and stocking fridge.
16	Person #2 and person #4 moving water cooler and water into area.
17	Cleaning person taking trash and foosball pilgrimage in background.
18	Eight individuals and two conversations
21	Person #5 acting suspicious, person #6 sliding on his back across floor, a chair moving across the floor on its own, a hand from behind curtains and behind chair and under table, person #6 with shirt over head
22	Person #2 in conversation and testing edge of visibility of system.
24	Person #1 and person #7 stocking fridge.
26	Person #8 (not from the building) retrieving the cans for recycling.
29	Person #7 giving demo to young man.
30	Three people in conversation.

Table 5.1: This table lists the top twenty periods of anomalous activity in the office environment. These periods represent less than .01% of the experiment. (Chris Stauffer was person #1 and Mike Ross was person #2.)

Chapter 6

Discussion and future work

The previous chapters have shown general mechanisms for learning and exploiting the structure of perceptual data. This chapter discusses the motivation and promise of this type of research.

Section 6.1 covers some of the qualitative motivation for this research. Section 6.2 begins to discuss avenues of future investigation related to or enabled by this research. This includes improvements in all the major components of the learning system as well as adding additional capabilities and supervision to the system. Supervision is useful in enabling a system to communicate its findings as well as allowing a system to refine its representations to better conform to what operators would expect. Finally, Section 6.3 covers factors that must be considered when applying these technologies to different application areas.

6.1 Discussion

Sometimes the reason for a research effort can get lost in the details. This section discusses some of the issues that motivated this research and will hopefully help motivate future research in this area. Subsection 6.1.1 discusses some basic motivation for Perceptual Data Mining. Subsection 6.1.2 casually relates some basic neural mechanisms to some of the mechanisms discussed in this document.

6.1.1 Motivation

By a modest age, a child has observed more data than the sum total of every experiment in computer vision in history. By some rough calculations, that age may be as young as one year. The child observes more "pixels" in a few seconds than the COIL-120 database commonly used for object recognition experiments. The child observes more "pixels" in a few minutes than the entire FERET database used for face recognition. The child has a continuous stream of visual input, not individual observations. Furthermore, that stream of input is highly structured.

The same child is capable of great understanding even before it has had many objects explicitly labeled for him/her. It is extremely adept at taking advantage

of implicit supervision available to it. This stands in stark contrast to much of the research in computer vision where most object detection or object classification problems are currently approached with a small corpus of explicitly labeled individual data points and, until recently, little unsupervised data.

The previous paragraph outlines the basic tenets of Perceptual Data Mining. First–do not try to learn with very little data. Out of necessity in the past, computer vision researchers collected a few images and manually labelled them. Learning from such limited amounts of this type of disembodied data is extremely difficult. Anecdotally, people say that humans can learn to recognize objects from a single image. While this statement is true, it must be qualified by "After experiencing more perceptual input than has ever been processed by any computer over years of an individual's life, that individual can ..." This qualification should be applied to most statements of what a human can do with limited information or training.

Second–use the information available in the continuous stream of data. A fundamental characteristic of all intelligent life is that it has exactly one perceptual data stream available from birth to death. While artificial systems may one day overcome this limitation, a majority of research in perception today involves data taken completely out of context. This disembodied data is extremely information poor. This thesis discussed some of the information available in a continuous stream of data over extended periods of time. It is not possible to prove that continuous data is required to develop intelligence, nor is it possible to prove otherwise. Until recently, the conventional computation, memory, and storage requirements for capturing and processing real-time video were prohibitive. In the past 5 years, real-time continuous visual processing has become a reality and enabled this line of research.

Humans innately develop the skill to stabilize moving objects. This is a preattentive process–not requiring active thought. At a very young age by saccading to an area of interest and stabilizing it, they could receive as input sequences of images of the same object as the object and its environment change. It is a strong belief of this researcher that this is a fundamentally enabling mechanism that allows a human visual system the ability to bootstrap from a primitive state. A significant part of this thesis centered on quantifying some of the sources of implicit supervision that can be exploited to create more effective models of the environment.

It is our hypothesis that the visual system cannot develop without continuous data. Unfortunately, this hypothesis is not provable without horrific experimentation, but psychophysiological studies on cat visual cortex have proven that without certain visual stimuli the feline visual system does not develop the ability to represent those stimuli properly. In particular, it lacks the ability to represent aspects of the environment which its visual system was deprived of experiencing. What role continuous visual stimuli play in the development of all known existing visual systems is not known. We acknowledge that there are other potential requirements for the development of intelligence that Perceptual Data Mining currently neglects. For instance, physical interaction and exploration may play an essential role in learning a powerful, composable representation of objects. This is an area we would like to investigate in the future.

The final tenet is, in active environments, a large amount of structure of a space

can be learned before supervision is used. Any supervised approach is fundamentally limited by the amount and type of supervision it can acquire. Of course, supervision and interaction are important in relaying information about what has been learned about an environment to other entities. But, beginning the process with explicit supervision is not conducive to a system that learns from very few examples and can adapt to changes in the task being performed or changing environments.

6.1.2 Biology Aside...

This section will not be satisfying to neuroscientists and is meant only to relate the elements of MOL to biological artifacts. Two major aspects of the Perceptual Data Mining system can be simply described as Hebbian learning on a competitive network. Lateral inhibition and Hebbian learning have been investigated by many biologically motivated researchers [64, 31].

Codebook generation as competitive learning

Given a set of abstract nodes, F, that represent the values observation features and a set of nodes, P, that respond to particular local values of them, the node p_{max} with the highest response can be adapted to represent its past data as well as the current observation. After experiencing enough data in this manner, our prototypes will become more representative and selective to the types of observations in the particular environment.

We are agnostic to the exact mechanism of competition between the responses of P and the model of each prototype nodes response. Our only requirement is that an observation vector is efficiently represented by one of a set of prototypes.

Co-occurrence learning as second order competitive learning

Given aggregate measures of our competitive responses, $\bar{P}$, an additional layer of abstract co-occurrence nodes, C, which respond to distributions of responses on P could be used to model latent classes. The node c_{max} which is most characteristic of the current prototype profile would be adapted to represent its past profiles as well as the current profile. While this is not as pleasing as the probabilistic model, it may help one understand how PDM could be implemented in a massively parallel system.

6.2 Future work

This thesis only scratches the surface of Perceptual Data Mining (PDM). This section discusses many future areas of research related to or enabled by PDM. While they each will add complexity to a PDM system, they all show promise for adding additional structure and constraints. This may make some seemingly insurmountable problems more tractable.

6.2.1 Attention improvements

Before developing the tracking system discussed in this thesis, we were incapable of doing any of this research. Once we had a tracking system that was robust enough to reliably track any object in multiple environments continuously for months, we could consider types of modeling that would not be reliable with sparse data. The ultimate tracking system would be a static attention based tracking solution that can track any object in any video source, but any improvements that result in more reliable and general tracking would result in greater applicability of the entire Perceptual Data Mining system. Although we find the results of the tracker encouraging, there are still opportunities for improvement.

As computers improve and parallel architectures are investigated, this algorithm can be run faster, on larger images, and using a larger number of Gaussians in the mixture model. All of these factors will increase performance. A full covariance matrix would further improve performance. Adding prediction to each Gaussian (e.g. the Kalman filter approach), may also lead to more robust tracking of lighting changes.

Beyond these obvious improvements, we are investigating modeling some of the inter-dependencies of the pixel processes. Relative values of neighboring pixels, correlations with neighboring pixel's distributions, and simple texture measures may be useful in this regard. This would allow the system to estimate changes in occluded pixels by observations of some of its neighbors.

Our method has been used on gray-scale, RGB, HSV, and local linear filter responses. But this method should be capable of modeling any streamed input source in which our assumptions and heuristics are generally valid. We are investigating use of this method with frame-rate stereo, IR cameras, and including depth as a fourth channel(R,G,B,D). Depth is an example where multi-modal distributions are useful, because while disparity estimates are noisy due to false correspondences, those noisy values are often relatively predictable when they result from false correspondences in the background.

Other potential improvements to our single camera system include automated setting of learning rates, disambiguation of lost tracking sequences using appearance and behavior models, multi-camera models for correspondence without visual overlap, and fast scene-wide context switching. If a robust, static attention mechanism could be bootstrapped, it may facilitate other application areas including database mining and web mining. To our knowledge, the idea of bootstrapping an attention mechanism from an active environment is novel.

6.2.2 Incorporating other Modalities

Any sensor can be directly incorporated into the system. Each sensor may introduce new regularities that can be used to represent the observations. There are many interesting questions regarding clustering using multiple sources of information.

Audio

Audio can tell a lot about the type of activity in an environment. Research in Computational Auditory Scene Analysis is centered on the problem of factoring an audio signal into the components made by different sources. Currently, most CASA systems run a few orders of magnitude slower than real-time, but simple approaches may work in the same types of sparse environments where PDM is currently applied.

A system that is capable of segmenting audio from different sources and clustering that audio could be directly incorporated into the PDM framework. This is an area that we intend to pursue.

Other sensors

Other sensors that could be used in a PDM system include door sensors, window sensors, weight pads, motion sensors, beam sensors, IR/Ultrasound tagging sensors, fingerprint systems, retinal scanners, face recognition systems, voice recognition systems, ignition sensors, equipment sensors, light sensors, wind/rain indicators, and temperature sensors. Learning relationships between these sensors and the other observations may result in a more robust system.

6.2.3 Transferable, factored representations

Our experiments were performed in single environments in which most of the actions performed were global. There was generally no need to segment the sequences to effectively represent them. If the object activities were combinations of other actions, a system that could determine a set of component actions to represent the complex actions would be a better model for the observations. For instance, segmenting pedestrian movement into walking, running, and standing movements may facilitate better understanding of pedestrian activities.

It would also be advantageous build a representation that can be used in multiple environments. Not only would such a system be capable of learning from the regularities it observes in *all* the environments, but it would be able to transfer latent class models and supervision to the other environments. For instance, if the size and velocities of objects can be regularized in multiple environments, the scalar factor which relates sizes in one environment to the other can be estimated. Thus classes based on size and velocity could transfer if the two scenes contain similar objects performing similar actions. This could also be used to measure the similarity of different scenes and cluster scenes which are functionally similar.

6.2.4 Additional context

We have shown how a period context cycle can be exploited to articulate a time-sensitive model of the objects and activities in a scene. Unfortunately, this will only work if the context cycles are periodic and regular. A conference room may have a very regular weekly schedule, but its schedule may include impromptu meetings, talks, conversations, and passing traffic. Using a timed Hidden Markov Model to

model different contexts of usage would allow a system to have an understanding of the major modes of use of an environment and a model of the expected activities during those contexts.

This context can also be exploited to model the environment. If the major context modes roughly correspond to different types of usage, the context state has an additional information about the environment. This context can be used to determine when unusual usage occurs. This is also true of the characteristics of the periodic context cycles. For instance, the timing of a traffic light often changes on a regular schedule. Two scenes that exhibit the same cycle period at the same times may be very likely to be related. Also, the existence and parameters of a salient period could be used to determine the type of scene (e.g., business, nature, traffic light, weekend getaway, etc.).

6.2.5 Supervision

We have argued that our approach to modeling the active elements of an environment is general. We have shown numerous different applications of our techniques to different aspects of perceptual data. We have shown the ability to build a robust, descriptive model of active elements of the environment, but to this point, our approach has not involved any supervision. Without some supervision our capabilities are limited to learning models of different types of activities, accumulating statistics of those activities, learning models for temporal context, and determining unusual activities under the model. This allows us to compress the data and determine outliers.

This section discusses what can be gained by adding supervision to the system. First, supervision as a means of evaluating the regularities captured by the model discussed. Then using supervision to improve the unsupervised model and control adaptive filtering of tracking events is covered. Finally, eliciting supervision and using that supervision to enable communication and interaction potentially through basic language is covered.

Analysis of supervision requirements

One method for evaluating a particular representation is looking at classification performance as a function of supervision. For some of the hierarchical models we have shown, a single labeled example from each class is likely to produce reasonable classification results. Given enough supervision, very simple classification mechanisms can perform optimally. Because of the high cost of supervision we would like our system to operate with minimal supervision.

To understand why so much effort was put into unsupervised data mining before supervision was used consider using the factored representation to solve multiple tasks. An example of some possible classification tasks in a parking garage environment are: vehicle type identification; vehicle color classification; vehicle activity classification; anomalous activity detection; etc. The information required for some of these tasks would be considered noise in the other tasks. Any or all of these tasks

might be useful in different environments. If supervision was used to guide any part of the process, the intermediate representation may not be of any use for other tasks.

In the future we will investigate properties of graphs of classification performance vs. supervision. One measure of classification performance is the equal error rate (EER)[1]. The supervision could be the number of randomly chosen training examples from each class. While most current research is concerned with achieving very good performance with large amounts of supervision, our goal is achieving reasonably good performance with little supervision (as humans do).

Backpropogating supervision

There is no reason supervision could not be used to improve our unsupervised representation. For instance, if a prototype contained two, mutually exclusive class labels, it could be split or the prototypes could otherwise be altered to increase their purity. Similarly, if two similar prototypes were functionally equivalent, they could be joined. Supervision which relates classes could also be applied (e.g., trucks and cars are both vehicles).

Supervision could also take the form of manually labeling pairs of observation as equivalent or different. This relational supervision could be useful in cases where exact class labels are not certain.

The system could also perform exploration. Rather than providing the system with labeled random examples, the system could ask the operator about particular examples whose class is uncertain. This is similar to how children tend to learn (e.g. "Daddy, what's that?" or "Is that a bird?").

Importance and typicality

While we have shown some ability to determine anomalous events, it is apparent that not all anomalies are interesting and some activities that are not anomalous may of interest to an operator. An operator could provide two types of supervision about tagged events–importance and typicality.

If an event occurs that is very similar to a latent class, it could be labeled as typical of that class. This information could be used to alter the representation to avoid future misclassification. If an event is labeled as atypical, it should not be incorporated into the model.

Regardless of the typicality of an event, an operator may or may not place importance on that event. For instance, a delivery may be atypical or typical of a particular environment but that fact does not bear on whether an operator wishes to be informed of that event. All typical activities can be easily reported or ignored.

By attempting to estimate these two properties of each incoming observation set, the most unusual and interesting events can quickly be reported. There may be situations where multiple sets of typicality or importance functions may be estimated for different user bases on the same system.

[1]It is possible to produce surface plots of ROC-curves vs. supervision, but it would be difficult to compare multiple such surface plots in a single figure.

Exploration and interaction

As stated earlier, interaction and exploration are important aspects of development. Our current system is completely passive. A system that can control the sensors, interact with active elements of the environment, control elements of the environment, or interact with the the supervision process could create a more effective model of the environment.

Lexicon mining and binding

Given text or speech segments that describe objects, parts of objects, activities, or locations, one could learn a lexicon which describes the observations, the binding of the lexicon to our representation, the grammar for that lexicon, and overall sentence level meaning. One could determine descriptive words and phrases (e.g. yellow, red, blue, black, jacket, shirt, shorts, pants, on the grass, on the street, blond, dark hair, person, man, woman, entering the building, leaving the garage, etc.). One could then bind these phrases to aspects of our representation while understanding that sets of phrases describe separate aspects (e.g. color vs. clothing) and that some phrases are synonyms with respect to our representation (e.g. coat, jacket, shirt).

Deb Roy [49] and Tim Oates[40] have attacked this problem with a bias towards mining the audio representation. Our automatically derived, rich representation would allow for complex sentences where the grammar becomes important. For instance, the sentence "a woman standing on the grass with a blue shirt and red pants." requires knowledge about which component regions the "blue" and "red" are bound to.

6.3 Applications

There are many potential application areas including security, elder-care, child-care, wildlife monitoring, home monitoring, traffic statistics, and intelligent environments. The breadth of these application areas and their conflicting goals accentuates the importance of Perceptual Data Mining.

6.3.1 Compression and scene statistics

The most straightforward application of Perceptual Data Mining is to record what is happening in an environment. While this could also be accomplished by a VCR, our representation can allow us to iteratively reduce the representation while maintaining most of the information.

Here, we will evaluate the size required to store the tracking information relative to storing the entire video stream. The size of the video stream is NFT, where N is the number of pixel in an image, F is the frames per second, and T is the length of the sequence. By not storing the background, we reduce the storage requirements by R, the relative number of pixels on moving objects in an average frame. This is approximately $\frac{nd}{N}$, where n is the average number of pixels on a moving object and

d is the "duty cycle" of an object, or the average number of objects on a particular frame[2]. Of course, compression can by applied to the video stream, but compression can be applied to the tracked objects as well. This is a lossy compression because the slow changes in the background are lost, therefore it is not possible to reconstruct a particular image exactly. Fortunately, for all the applications we have considered, the appearance of the background is usually of little importance in understanding what is happening.

While this is a significant reduction in storage requirements, it is more interesting to consider what can be gained using the statistical model we have built. For instance, if a billion individuals walk across the same path, their object type, appearance, and activities can be specified parametrically on the model we have estimated. This can result in more than two orders of magnitude reduction in size of the representation. This not only allows for massive reduction in storage requirements, but enables data mining tasks that would not be feasible without such a reduction. While very subtle variations may be lost, this is appropriate for many of the applications we are considering.

6.3.2 Anomaly detection

For many applications the unusual activities are of most interest. E.g., a vehicle speeding towards an embassy, a grandmother falling in the bathroom, a child climbing a bookshelf, an accident on a highway, the rare tiger, etc. Unusual patterns of activities can also be of interest. E.g., deliveries at unusual times, vehicles running traffic lights, no traffic during morning rush hour, a child getting up in the middle of the night.

In the future, other types of anomalies could be modeled. Long-term variations could be of interest. E.g., this year Edna gets up one hour later on average; etc.

[2]Note that this can be greater than 1.

Chapter 7

Conclusions

The Perceptual Data Mining (PDM) framework introduced in this thesis shows promise in bootstrapping perceptual intelligence. Given a primitive system that detects the presence of active objects, the PDM system can track objects in multiple sensors, build models of object shape, build models of activities, build models of object appearance, build context sensitive models for a given environments, and determine events that are uncharacteristic of the environment.

The primary contribution of this thesis was to motivate this problem and to advocate our bottom-up, data-driven framework. Our framework exploits different regularities of real-world environments in a particular order to build an extremely rich model of the active objects of an environment. As shown by our broad range of results, we have achieved this goal while maintaining a generally applicable system.

Secondary contributions include each individual component of the Perceptual Data Mining framework. The most essential is the adaptive background mixture model. This general background modeling technique enabled robust tracking of a wide range of objects in a variety of difficult environments. The presence of this tracker and the data it produced motivated this research. Our general correspondence framework incorporated both immediate and continuous correspondence and shows promise for long-term correspondence.

Multiple Observation Learning (MOL) proved to provide robust, probabilistic models of many different phenomena including shape clusters, activity clusters, pixel region clusters, and pixel value clusters. A particular contribution of this work was the adaptation and analysis of this type of co-occurrence method to continuous observation spaces. Different methods of representing the continuous observations were investigated including Associative Mixtures of Gaussians (AMG) which incorporates our latent class estimates to find a representation of the data which has low entropy with the latent class.

The strength of Multiple Observation Learning was accentuated by the context-based analysis and anomaly detection that it enabled. We were able to determine periodic cycles of activity in multiple environments. We were also able to exploit these context models to increase our capability of anomaly detection.

The most intriguing aspect of the Perceptual Data Mining framework is the potential for future investigation and application to real-world problems. Potential applica-

tions for the Perceptual Data Mining framework have only begun to be investigated. Improvements in attention and correspondence could allow these techniques to be applied to everything from personal video to broadcast video. Other modalities will enable greater depth in the representations built and modeling of interdependencies between modalities. More general, factored representations will enable more articulated description of the objects and activities. Supervision would enable directed adaptation of exploration, faster learning of more difficult concepts, and communication.

Appendix A

An efficient, approximate tracker implementation

This appendix outlines an efficient, *approximate* version of the single camera tracking system described in this document.

A.1 Outer loop

Our tracking system involves four components. These component are background estimation, connected components, continuous on-line tracking, and storage of tracking data. The outer processing loop performed for every frame is

1. **Background estimation**–Determine the foreground pixels.
2. **Connected Components**–Group pixels into "connnected" regions.
3. **Continuous on-line tracking**–Track individual objects from frame to frame.
4. **Storage of tracking data**–Store instances of tracked object each frame.

The following sections describe different aspects of each of the major components of the tracking system. Each describes how to process an single step corresponding to a captured frame.

A.2 Background estimation

For each pixel in the frame...

1. Find the *first*[1] mixture component that "matches" (is within $\tilde{\sigma}$ of the current pixel observation).

[1]The ordering of the components is described later.

2. If the no match is found, replace the *last* mixture component with a new component centered at the current observation with a large variance and small standard deviation. This is the new match

3. Determine whether the corresponding mixture component is likely to be a foreground component. If the combined weights of the previous checked components exceeds a threshold, it is a foreground component.

4. Update the parameters of the components. Increase the weight of the current component relative to the previous components and adjust the mean and standard deviation of *only* the matching component.

5. Reorder the components such that the components that are most likely to be background are first on the list. We use the value $\frac{w}{\sigma}$ to order the components. This optimizes the search such that most matches take one test.

A.2.1 Notes on adaptive backgrounding

Above is a concise restatement of the information in the previous chapters, but there are some details that still need to be discussed. Starting from the beginning, the value of $\tilde{\sigma}$ is dependent on the noise of the camera and the environment. A value of 2.5σ is fairly robust. If the value is too small, the estimated variance mixture component may decrease until a singularity is reached. If the value is too large, foreground pixel values may be assigned to the wrong component and corrupt the model. In general that threshold should be set so there is a small amount of "snow" in the foreground image. We have considered employing this heuristic to adapt this threshold.

If a match is not found the new component should be initialized at the location of the current value and the variance should be initializes as larger than a standard component. If the variance is too large, it will take a large number of frames to decrease. If it is too small, it may not represent the new pixel observations. Using a variance that is 2-5 times that of an average adapted component is reasonable.

Updating the components also introduces some complexity. This document described two adaptation thresholds for the weight and the parameters (σ and μ) of the mixture component. In general, the weight should be adapted more slowly than the parameters of the component. In fact, it is possible to adapt the mean faster than the variance to allow for better tracking. It is often advantageous to enforce a minimum variance. By defining your adaptation coefficient in terms of seconds rather than frames, these values can be approximated when the framerate is variable.

A.3 Connected Components

Given a foreground image,

1. Pass over the pixel in raster order. If the pixel is a foreground pixel, check for previously visited pixels within its connection region (usually defined by a radius) for other (labelled) foreground pixels.

(a) If no foreground pixels are present, label this foreground pixel with a new unique label.

(b) If one foreground pixel is present, label this pixel with its unique label.

(c) If more than one foreground pixel is present, label this pixel with the first label of the first match and record the equivalency of all pixel pairs.

2. Pass over the pixel in raster order again labeling each foreground pixel with the smallest equivalent unique label. This is also a good time to record sufficient statistics for moments and bounds of the connected components.

A.3.1 Notes

Bytes are often not sufficient as unique labels. It is important to filter connected components that are smaller than a few pixels because the background estimation technique is designed to produce a small amount of white noise.

A.4 Continuous on-line tracking

We have not advocated any particular type of tracking. Thus, we will not elaborate on what was described in Chapter 3 here.

A.5 Storage of tracking data

Storing completed tracking sequences is more reliable than storing each instance of each frame. This is because more complex tracking system may alter their unique labels. For each object in each frame, one can store the image, the foreground image, the position, velocity, size, and even the depth of pixels if available.

Appendix B

Estimating homographies

Two corresponding points are related by a homography transformation as follows.

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \propto \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \tag{B.1}$$

To solve for a homography given a set of point pairs, one can represent the three constraints for each pair of points as three rows row as follows.

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & 0 & 0 & -x_0' & 0 & \dots & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & 0 & 0 & -y_0' & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_0 & y_0 & -1 & 0 & \dots & 0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -x_1' & \dots & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & 0 & 0 & 0 & -y_1' & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & 0 & -1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_N & y_N & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & -x_N' \\ 0 & 0 & 0 & x_N & y_N & 1 & 0 & 0 & 0 & 0 & \dots & -y_N' \\ 0 & 0 & 0 & 0 & 0 & 0 & x_N & y_N & 0 & 0 & \dots & -1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ -1 \\ \dots \\ \dots \\ \dots \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{B.2}$$

$$a * b = c \tag{B.3}$$

(B.4)

If the pairs are weighted into the estimation, a weight matrix can also be computed as

$$W = \begin{bmatrix} w_1 & 0 & 0 & 0 & 0 & \dots \\ 0 & w_1 & 0 & 0 & 0 & \dots \\ 0 & 0 & w_1 & 0 & 0 & \dots \\ 0 & 0 & 0 & w_2 & 0 & \dots \\ 0 & 0 & 0 & 0 & w_2 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \tag{B.5}$$

Thus, the homographies and the alpha values can be computed

$$b = (Wa)^{-1} * Wc. \tag{B.6}$$

Appendix C

Factorizing joint distributions

Given the estimated distribution, C, and initial estimates of the weights and class-conditional densities in matrix form (W and P_c) such that W is an NxN matrix with class weights along the diagonal and P_c is an NxK matrix in which each row corresponds to the pmf of one class. Hence, $\hat{C} = P_c' * W * P_c$.

1. Compute estimated co-occurrence matrix. For example in Matlab,
   $\hat{C} = P_c' * W * P_c$.
2. Compute estimated element-wise relative error[1].
   $E = C./\hat{C}$;
3. Re-estimate the parameters of the model.
   $P_c^{new} = P_c. * (P_c * E)$
   $W^{new} = W. * (P_c * E * P_c')$
4. Renormalize weights and conditional pmfs of latent classes.
   $W = W/sum(sum(W))$
   $P_c = P_c./repmat(sum(P_c, 2), [1, K])$

C.0.1 Notes

An attempt has been made in this thesis to include aspects of data normalization (see Chapter 4. In some cases, there is little mention of normalization of co-occurrence measurements. For instance, document-word co-occurrence matrices are often normalized such that each document represents an equal amount of co-occurrence. Also, word co-occurrences can similarly normalized, de-emphisized, or removed if the words are likely to be discriminative or likely to be uninformative.

Also, estimated co-occurrence matricies are often sparse. A zero in the co-occurrence matrix can cause computation problems.

[1]If estimated conditional probabilities reach "zero" within the precision of the machine, some elements of this matrix will be invalid. Invalid elements can be assigned the value of one.

Bibliography

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

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision (ICCV99)*, September 1999.

[3] Robert Collins, Alan Lipton, and T. Kanade. A system for video surveillance and monitoring. In *Proc. American Nuclear Society (ANS) Eighth International Topical Meeting on Robotic and Remote Systems*, pages 25–29, Pittsburgh, PA, April 1999.

[4] S. Coorg and S. Teller. Extracting textured vertical facades from controlled close-range imagery. In *Proceedings of CVPR*, pages 625–632, 1999.

[5] CA Corel Corporation, Ontario. Corel stock photo library. Available from Corel Corp., 1990.

[6] T. Darrell, G. Gordon, J. Woodfill, and M. Harville. A virtual mirror interface using real-time robust face tracking. In *Proceedings of the the Third International Conference on Face and Gesture Recognition.* IEEE Computer Society Press, April 1998.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39((Series B)):1–38, 1977.

[8] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In D. Vernon, editor, *Proc of the European Conference on Computer Vision (ECCV 2000)*, pages 751–767. Springer Verlag, 2000.

[9] T. Ellis and M. Xu. Object detection and tracking in an open and dynamic world. In *Proc. of the 2nd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2001)*, Kauai, Hawaii USA, December 9-14 2001.

[10] Larry Davis et al. Visual survellance of human activity. In *Asian Conference on Computer Vision (ACCV98)*, Mumbai, India, January 1998.

[11] Michael J. Evans, Zvi Gilula, and Irwin Gittman. Latent class analysis of two-way contingency tables by bayesian methods. In *Biometrika*, volume 76(3), pages 557–563, September 1989.

[12] O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In *Computer Vision - ECCV'92, Lecture Notes in Computer Science*, volume 588, pages 563–578. Springer-Verlag, 1992.

[13] M. Fischler and R. Bolles. Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[14] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Proc. of the Thirteenth Conference on Uncertainty in Artificial Intelligence(UAI)*, August 1-3 1997.

[15] Gersho and Gray. *Vector Quantization and Signal Compression.* Kluwer Acedemic Press, 1991. ISBN 0-7923-9181-0.

[16] Ismail Haritaoglu, David Harwood, and Larry S. Davis. w^4: Who? when? where? what? a real time system for detecting and tracking people. In *Third International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, April 14-16 1998.

[17] Ismail Haritaoglu, David Harwood, and Larry S. Davis. A fast background scene modeling and maintenance for outdoor surveillance. In *Proc. of the Internactional Conference on Pattern Recognition (ICPR00)*, pages 179–183, Barcelona, Spain, September 3-8 2000.

[18] R. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Computer Vision - ECCV'92, Lecture Notes in Computer Science*, volume 588, pages 579–587. Springer-Verlag, 1992.

[19] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, San Francisco, 1999. Morgan Kaufmann Publishers, Inc.

[20] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[21] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[22] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. In *PAMI*, volume 15, pages 850–863, 1993.

[23] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proceedings of the 4th International Conferenceon on Computer Vision*, pages 93–101, 1993.

[24] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proc of the European Conference on Computer Vision (ECCV 2000)*, pages 343–356. Springer Verlag, 1996.

[25] Yuri Ivanov, Aaron Bobick, and John Liu. Fast lighting independen background subtraction. Technical Report 437, MIT, Media Laboratory, 1997.

[26] Omar Javed, Sohaib Khan, Zeeshan Rasheed, and Mubarak Shah. Camera handoff: Tracking in multiple uncalibrated stationary cameras. In *Proceedings of the Workshop on Human Motion (HUMO'00)*, pages 113–119, 2000.

[27] Neil Johnson and David C. Hogg. Learning the distribution of object trajectories for event recognition. In Pycock D., editor, *British Machine Vision Conference (BMVA)*, pages 583–592, September 1995.

[28] N. Jojic and B. J. Frey. Topographic transformation as a discrete latent variable. In T. K. Leen S. A. Solla and K.-R. Muller, editors, *Advances in Neural Information Processing Systems 12*, Cambridge, MA, 1999. MIT Press.

[29] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Kauai, HA, Dec 9-14 2001. MIT Press.

[30] V. Kettnaker and R. Zabih. Bayesean multi-camera surveillance. In *Computer Vision and Pattern Recognition (CVPR 1999)*, pages 253–259, Fort Collins, Colorado USA, June 23-25 1999.

[31] Teuvo Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, Germany, third edition, 1989.

[32] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel. Towards robust automatic traffic scene analysis in real-time. In *Proc. of the International Conference on Pattern Recognition*, November 1994.

[33] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[34] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Adv. Neural Info. Proc. Syst. 13 (NIPS 2000)*, pages 556–562, 2001.

[35] Lily Lee, Raquel Romano, and G. Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *PAMI*, 1999. separate paper in this issue.

[36] Alan Lipton, Hironobu Fujiyoshi, and Raju S. Patil. Moving target classification and tracking from real-time video. In *IEEE Workshop on Applications of Computer Vision(WACV)*, pages 8–14, Princeton NJ, October 1998.

[37] J. P. Mellor. Reconstructing built geometry from large sets of calibrated images. Technical Report AITR-1674, MIT, October 1999.

[38] E. Miller, N. Matsakis, and P. Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2000)*, volume 1, pages 464–471, 2000.

[39] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-100). Technical Report CUCS-006-96, Columbia University, February 1996.

[40] Tim Oates, Zachary Eyler-Walker, and Paul R. Cohen. Toward natural language interfaces for robotic agents: Grounding linguistic meaning in sensors. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 227–228, 2000.

[41] Nuria Oliver, Barbara Rosario, and Alex Pentland. A bayesian computer vision system for modeling human interactions. In *Proceedings of ICVS99*, Gran Canaria, Spain, January 1999.

[42] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognition*, pages 193–199, Puerto Rico, June 16-20 1997.

[43] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, pages 555–562, January 1998.

[44] F.C. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio, 1993.

[45] P. J. Phillips, P. Rauss, and S. Der. Feret (face recognition technology) recognition algorithm development and test report. Technical Report ARL-TR-995, U.S. Army Research Laboratory, 1996.

[46] T. Poggio and K.K. Sung. Example-based learning for view-based human face detection. In *Proc. of the ARPA Image Understanding Workshop*, volume 2, pages 843–850, 1994.

[47] A. L. Ratan. Training templates for scene classification using a few examples. In *Proc. of the IEEE Content Based Access of Image and Video Libraries*, San Juan, 1997.

[48] Christof Ridder, Olaf Munkelt, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on recent Advances in Mechatronics (ICRAM95)*, pages 193–199, 1995. UNESCO Chair on Mechatronics.

[49] Deb Roy. *Learning from Sights and Sounds: A Computational Model.* PhD dissertation, Massachusetts Institute of Technology, Media Lab, June 1999.

[50] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[51] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proc. of the IEEE Conf on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997.

[52] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997.

[53] Chris Stauffer. Automatic hierarchical classification using time-based co-occurrences. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR1999)*, Fort Colins, CO, June 1999.

[54] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition (CVPR99)*, Colordo Springs, CO, June 1999.

[55] Chris Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8), August 2000.

[56] Gideon Stein. Tracking from multiple view points: Self-calibration of space and time. In *Image Understanding Workshop*, November 1998.

[57] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proceedings IEEE Int. Conf. on Computer Vision*, pages 255–261, Corfu, Greece, September 1999.

[58] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[59] P. Viola. *Alignment by Maximization of Mutual Information.* PhD dissertation, MIT, Artificial Intelligence Lab, June 1995.

[60] Paul Viola and Mike Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of the Conf. On Computer Vision and Pattern Recognition*, 2001.

[61] W.E.L.Grimson, Chris Stauffer, Raquel Romano, and Lily Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. of Computer Vision and Pattern Recognition (CVPR98)*, Santa Barbara, CA, June 1998.

[62] Mike Wessler. A modular visual tracking system. Masters thesis, MIT, Artificial Intelligence Lab, June 1995.

[63] Christopher R. Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[64] X.-H. Xie, R. Hahnloser, and H. S. Seung. Learning winner-take-all competition between groups of neurons in lateral inhibitory networks. In *Adv. Neural Info. Proc. Syst. 13*, pages 350–356, 2001.

[65] L. Zelnik-Manor and M. Irani. Multi-frame estimation of planar motion. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, volume 22(10), pages 1105–1116, October 2000.