LEARNING ON RIEMANNIAN MANIFOLDS FOR INTERPRETATION OF VISUAL ENVIRONMENTS

BY CUNEYT ONCEL TUZEL

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Peter Meer

and approved by

New Brunswick, New Jersey

October, 2008

ABSTRACT OF THE DISSERTATION

Learning on Riemannian Manifolds for Interpretation of Visual Environments

by Cuneyt Oncel Tuzel

Dissertation Director: Peter Meer

Classical machine learning techniques provide effective methods for analyzing data when the parameters of the underlying process lie in a Euclidean space. However, various parameter spaces commonly occurring in computer vision problems violate this assumption. We derive novel learning methods for parameter spaces having Riemannian manifold structure and present several practical applications for scene analysis.

The mean shift algorithm on Lie groups is a generalization of the mean shift procedure which is also an unsupervised learning technique for vector spaces. The derived procedure can be used to cluster data points which form a matrix Lie group. We present an application of the new algorithm for multiple 3D rigid motion estimation problem from noisy point correspondences in the presence of outliers. The approach performs simultaneous estimation of all the motions and does not require prior specification of the number of motion groups.

We present a novel geometric framework to learn a supervised classification model for data points lying on a connected Riemannian manifold. The structure of the classifier is an additive model, where the weak learners are trained on the tangent spaces of the manifold. The derived algorithm is applied to pedestrian detection problem which is known to be among the hardest examples of the detection tasks.

We describe a regression model where the response parameters form a Lie group. The model is utilized for affine tracking problem where the motion is estimated as a parameter of the image observations. We present generalization of the learning model to build an invariant object detector from an existing pose dependent detector. The proposed model can accurately detect objects in various poses, where the size of the search space is only a fraction compared to the existing detection methods.

The other contributions of the thesis include a novel region descriptor and an online learning algorithm for estimating background statistics of a scene which are utilized for several challenging problems such as matching, tracking, texture classification and low frame rate tracking.

Acknowledgements

I would like to express my deepest gratitude to my advisor Peter Meer for his invaluable guidance over the course of time I was in Rutgers. His enthusiasm and integral view on research has made a deep impression on me. I also would like to thank Peter and Ioana Meer for their hospitality and friendship.

Many thanks to Fatih Porikli for his advice, encouragement, and support over the years. He has been an excellent mentor on every aspect of my life.

I want to express my appreciation to my colleagues from the Robust Image Understanding Laboratory, Raghav Subbarao, Lin Yang, Haifeng Chen, and Sushil Mittal for their friendship and many insightful discussions. I also thank all my colleagues who turned the years at graduate school into a pleasant time.

I am very grateful to my parents Sule and Hasan Tuzel, and my sister Selale Tuzel for their unconditional support and love at every step of my life.

Above all, I thank to my dear wife Ozlem for her love, support and patience.

Dedication

To my dear wife Ozlem.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The goal of the computer vision research is to build systems which can interpret the properties of the 3D world from the visual data. Since the problem is too large to conquer within a single framework, the task is divided into several conceptually meaningful subproblems. Each of the subproblems might require different interpretation of the measurements and the reliability of the approach depends on the robustness of the individual modules and ability to operate on noisy measurements.

The main challenge is that even the most primitive cognitive tasks are difficult to imitate using computers. For instance, understanding the 3D geometry of a scene from a video or recognizing a face given an exemplar image are fairly simple tasks for a human. However, manually specifying each step of the algorithms for the same tasks is far from realistic. Expert knowledge of the problem domain is not enough to come up with heuristic methods which can work in wide range of conditions and hand tuning the parameters of the system. A more promising approach is to present a data driven solution such that the machine learns and refines the algorithm using the available data.

With the improved computational power of the computers, the advances in machine learning techniques, and the availability of the labeled datasets, learning based methods become increasingly popular in computer vision research. Machine learning techniques have been successfully used in variety of computer vision problems from low level tasks such as interest point detection and feature extraction, to mid level tasks such as matching and motion estimation, to high level tasks such as tracking, object detection and event recognition.

A limitation of the classical machine learning techniques is the assumption that the data points form a vector space. However, several parameter spaces commonly occurring in computer vision problems are non-Euclidean in nature. These spaces include smooth, curved surfaces embedded in higher dimensional Euclidean spaces called manifolds. Consequently, the learning methods should be revised to handle the intrinsic geometry of the underlying space. For instance, it's well known that the Euclidean distance is not the most appropriate metric or the mean of the points in the Euclidean sense is not necessarily contained in the space. These and similar operators should be replaced with the equivalent forms defined on the manifolds.

In this thesis we present novel learning methods for a class of smooth manifolds which are equipped with a well defined notion of distance between the points, named Riemannian manifolds. The derived learning algorithms are utilized for several practical computer vision applications where the domains of the problems possess Riemannian manifold structure.

This thesis is organized as follows. In Chapter 2, we describe an unsupervised learning method for data points which form a Lie group, a special case of a smooth manifold which also has group structure. The original mean shift algorithm over Euclidean space is a well studied nonparametric clustering procedure which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. We derive an extension of the mean shift procedure where the parameter spaces form a Lie group. The algorithm is utilized to estimate multiple 3D rigid motions from noisy point correspondences in the presence of outliers. Unlike the existing techniques, the method does not require prior specification of the number of motion groups and estimates all the motion parameters simultaneously.

In Chapter 3, we describe a new descriptor based on the covariance matrix of image statistics computed inside a region of interest. Using a tensor of integral images, we derive a fast method to compute the covariance descriptor of an arbitrary rectangular region invariant of its size. The covariance descriptors do not form a vector space. However, the space of d-dimensional nonsingular covariance matrices (space of symmetric positive definite matrices) can be represented as a connected Riemannian manifold. We present three applications of the descriptors for region matching, texture classification and tracking utilizing nearest neighbor search based on an affine invariant metric defined for the space of symmetric positive definite matrices.

In Chapter 4, we describe a supervised learning algorithm for classifying data points lying on a Riemannian manifold. The structure of the classifier is an additive model where the weak learners are trained on the tangent spaces of the manifold. We show that the proposed logarithm charts minimize the approximation errors to the distances computed on the manifold. We present an application of the derived algorithm for the pedestrian detection problem in still images where the covariance descriptors are utilized as object features.

In Chapter 5, we present a novel learning based tracking model combined with object detection. The existing techniques proceed by linearizing the motion, which makes an implicit Euclidean space assumption. We describe a regression model learned on the Lie algebra of the transformation group and show that the formulation minimizes a first order approximation to the geodesic error. The learning model is extended to train a class specific tracking function, which is then integrated to an existing pose dependent object detector to build a pose invariant object detection algorithm. The proposed model can accurately detect objects in various poses, where the size of the search space is only a fraction compared to the existing object detection methods.

In Chapter 6, we describe an online learning approach to estimate the background statistics of a dynamic scene. The color distribution of the scene background is modeled with layers of normal distribution, and posterior density of mean and covariance are estimated via recursive Bayesian learning. The learned background statistics are then integrated to mean shift tracker and a robust low frame rate object tracker is presented.

In Chapter 7, we describe a decision support system to distinguish among hematology cases directly from microscopic specimens. The system uses an image database containing digitized specimens from normal and four different hematologic malignancies. Initially, the nuclei and cytoplasmic components of the specimens are segmented using a robust color gradient vector flow active contour model. Using a few cell images from each class, the basic texture elements (textons) for the nuclei and cytoplasm are learned, and the cells are represented through texton histograms. We propose to use support vector machines on the texton histogram based cell representation and achieve major improvement over the commonly used classification methods in texture research.

Conclusions and directions for future research are presented in Chapter 8. In Appendix A, we present a brief introduction to Riemannian geometry.

Contributions of the Thesis

Several main contributions of the theses are listed below.

- Generalization of mean shift algorithm to Lie groups. The new algorithm is valid over any matrix Lie group.
- Application of the derived mean shift algorithm for multiple 3D rigid motion estimation problem.
- A novel region descriptor referred to as region covariance descriptor. An $\mathcal{O}(d^2)$ algorithm is described to compute the covariance descriptor of an arbitrary rectangular region invariant of its size, where d is the dimension of the covariance matrix.
- Several applications of the region covariance descriptors utilizing nearest neighbor search based on an affine invariant metric defined for the space of symmetric positive definite matrices for
  - region matching
  - texture classification
  - tracking.
- A novel learning algorithm for classifying data points lying on a Riemannian manifold. The new algorithm is valid over any connected Riemannian manifold.
- Application of the derived learning algorithm for pedestrian detection problem in still images.
- A novel formulation for motion estimation by learning a regression model on the Lie algebra of the transformation group. The regression model is valid for any transformation group having matrix Lie group structure.

- Applications of the derived regression model for
  - affine tracking
  - invariant object detection.
- An online learning approach to estimate the background statistics of a dynamic scene.
- Low frame rate tracking application via integrating background statistics to mean shift tracker.
- A decision support system to distinguish among hematology cases directly from microscopic specimens based on a novel texton histogram representation and support vector machine classification.

Chapter 2

Clustering on Lie Groups

2.1 Introduction

Mean shift is an iterative procedure for locating the stationary points of a density function represented by a set of samples. Although the procedure was initially described decades ago [56], it's not been popular in vision community until its potential uses for feature space analysis and optimization were understood [23, 29]. Recently, the mean shift procedure is used for a wide range of computer vision applications such as visual tracking [11, 25, 43, 61, 65], image smoothing and segmentation [27, 175, 168], and information fusion [22, 26]. In addition, the theoretical properties of the procedure such as order and guarantee of convergence are discussed in several studies [18, 27, 45].

Mean shift clustering is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. The data points are assumed to be originated from an unknown distribution which is approximated via kernel density estimation. The cluster centers are located by the mean shift procedure and the data points associated with the same modes produce a partitioning of the feature space.

A limitation of the original mean shift procedure is that the data points are restricted to lie on a vector space. However, several important parameter spaces which commonly occur in computer vision problems do not form a vector space. Here, we derive an extension of the mean shift procedure where the parameter space forms a Lie group, a special case of a smooth manifold which also has group structure. We present an application of the derived procedure for the rigid motion estimation problem where the space of 3D rigid transformations form a matrix Lie group.

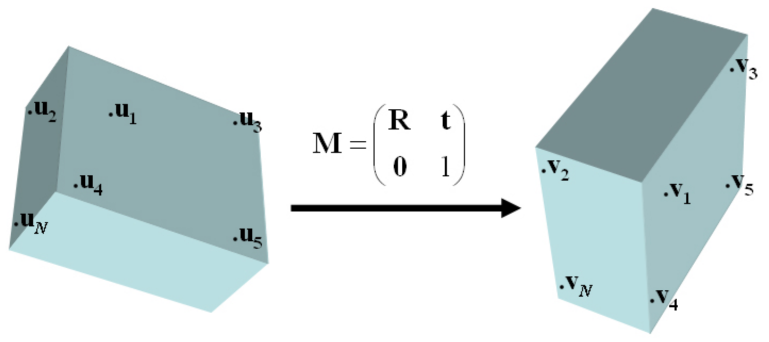
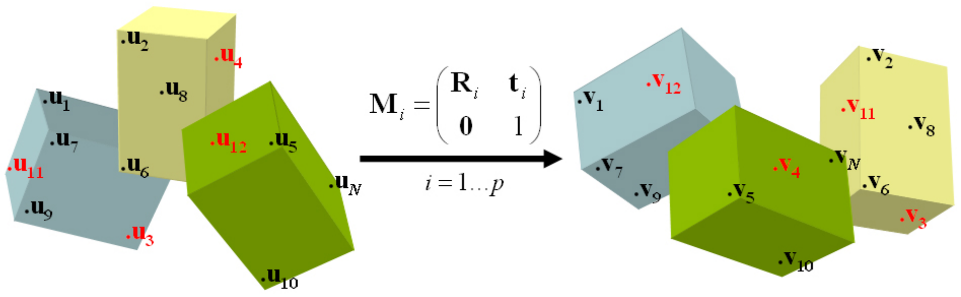


Figure 2.1: 3D rigid motion estimation. The two sets of matched points $U = \{\mathbf{u}_j\}_{j=1\ldots N}$ and $V = \{\mathbf{v}_j\}_{j=1\ldots N}$ are related through rotation $\mathbf{R}$ and translation $\mathbf{t}$.

Rigid motion estimation is a fundamental problem in computer vision. Given two sets of 3D points in correspondence, the aim is to find the rotation $\mathbf{R}$ and translation $\mathbf{t}$ parameters (Figure 2.1). The most popular techniques treat the problem as two sequential subproblems, estimation of rotation followed by estimation of translation. In [4, 160] two data sets are centered and rotation is estimated by singular value decomposition (SVD). Using the estimated rotation, translation is estimated with least squares solution. In a similar approach [75], quaternions are used to recover the rotation parameters. Experiments with synthetic data show that the two methods yield the same solution [42].

The methods mentioned assume that the data is corrupted with homogenous and isotropic noise. This assumption is not correct in general. Usually, 3D points come either from stereo pairs or range images. It is well known that noise along the depth direction is greater than along other directions [13]. Moreover if 3D measurements are recovered through triangulation process in a calibrated stereo configuration, points will have heteroscedastic errors. In the absence of translation, unbiased rotation can be recovered using the renormalization technique discussed in [118]. More recently, in [104], the authors proposed a solution in existence of both translation and rotation. The motion parameters are estimated by solving a heteroscedastic, multivariate errors in variables regression problem (HEIV).

Multiple motion estimation (Figure 2.2) is a much harder problem and none of



$\mathbf{M}_i = \begin{pmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{pmatrix}$
$i = 1 \dots p$
$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ $\mathbf{u}_4$ $\mathbf{u}_5$ $\mathbf{u}_6$ $\mathbf{u}_7$ $\mathbf{u}_8$ $\mathbf{u}_9$ $\mathbf{u}_{10}$ $\mathbf{u}_{11}$ $\mathbf{u}_{12}$ $\mathbf{u}_N$
$\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$ $\mathbf{v}_4$ $\mathbf{v}_5$ $\mathbf{v}_6$ $\mathbf{v}_7$ $\mathbf{v}_8$ $\mathbf{v}_9$ $\mathbf{v}_{10}$ $\mathbf{v}_{11}$ $\mathbf{v}_{12}$ $\mathbf{v}_N$

Figure 2.2: Multiple 3D rigid motion estimation. The points are either related through one of the p rigid motions $\mathbf{M}_i$ or they are outliers. The black and red points are the inlier and outlier points respectively.

the above methods can be used directly. The problem can be considered as estimating motion parameters in the presence of structured outliers. Although there is not much previous work done for estimating multiple 3D rigid motions, several studies are performed for estimating motion groups on 2D images [6, 170, 177].

The two most common approaches to 2D motion estimation are based on expectation maximization (EM) and iterative estimation of motions. In EM-based methods, point-motion association is followed with parameter estimation recursively until satisfactory results achieved. In iterative estimation of motions, the most dominant motion is detected considering points from other motions to be outliers. The outliers are usually found with random sample consensus (RANSAC) algorithm. Points from the detected motion are removed and the process is iterated. Both EM-based and iterative approaches require prior specification of number of motions. Moreover RANSAC algorithm requires an auxiliary scale parameter and in the original implementation the number of inliers should be more than the number of outliers. More recent approaches focus on motion clustering. In [89], tensor voting is used to cluster motion groups and estimate 2D motion parameters. In [164], 3D motions are detected in a noise free environment via clustering 2D point matches according to fundamental matrix constraints.

We present a novel approach to estimate multiple rigid motions from noisy 3D point correspondences in the presence of large amount of outliers [158]. Initially we do not reason about the point-motion associations. We find all the motion parameters simultaneously based on sampling and mode finding on the sampled distribution. The proposed method is robust and does not require prior specification of number of motions. Our approach has two major steps. The first step is to sample elemental subsets from the existing point matches and estimate the motion parameters. Motion estimation can be performed via any of the 3D rigid motion estimation methods discussed above. Finally, we find the modes of the generated rigid motion distribution using the derived mean shift procedure on Lie groups. Point-motion associations can be found with a simple post processing step.

2.2 Mean Shift on Vector Spaces

Here we present the derivation of mean shift procedure on vector spaces. Given n data points $\mathbf{x}_i$, $i = 1, ..., n$ on an m-dimensional space $\mathbb{R}^m$, the multivariate kernel density estimate obtained with kernel $K(\mathbf{x})$ and window radius h is given by

$$f(\mathbf{x}) = \frac{1}{nh^m} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right). \tag{2.1}$$

For radially symmetric kernels, it suffices to define the profile of the kernel, $k(x)$, satisfying

$$K(\mathbf{x}) = c_{k,m} k(\|\mathbf{x}\|^2) \tag{2.2}$$

where $c_{k,m}$ is a normalization constant which assures $f(\mathbf{x})$ integrates to one. Taking the gradient of (2.1), we observe that the stationary points of the density function satisfy

$$\frac{2c_{k,m}}{nh^{m+2}} \sum_{i=1}^{n} (\mathbf{x} - \mathbf{x}_i) g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) = 0 \tag{2.3}$$

where $g(x) = -k'(x)$. The solution can be found iteratively via the fixed point algorithm

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}. \tag{2.4}$$

At each step of the procedure a local weighted mean is computed. In the next iteration the computation is repeated centered on the previous estimate. The difference between the current and the previous location estimates is called the *mean shift vector*

$$\mathbf{m}_h(\mathbf{x}) = \bar{\mathbf{x}} - \mathbf{x}. \tag{2.5}$$

Comaniciu and Meer [27] show that the convergence to a local mode of the distribution is guaranteed when the mean shift iterations are started at a data point. See [27] for more details.

2.3 Mean Shift on Lie Groups

The mean shift algorithm presented in the previous section is not directly applicable to parameter spaces which do not form a vector space. For example, the mean shift procedure requires computation of the weighted mean of the points (2.4). Since the parameter space is not a vector space, the computed arithmetic mean is not necessarily contained in the space. Secondly, the Euclidean distance is not a proper metric to measure the similarity between the points.

Here we present a generalization of the mean shift procedure for parameter spaces having matrix Lie group structure. A Lie group is a group G which also has smooth manifold structure such that the group operations are analytic maps. For any smooth manifold, the local neighborhood of a point $\mathbf{X}$ can be described by its tangent space $T_{\mathbf{X}}$ which is a vector space. The tangent space to the identity element of the group $T_{\mathbf{I}}G$ forms a Lie algebra which is denoted by $\mathfrak{g}$. Matrix Lie groups are all the subgroups of the general linear group $\mathbf{GL}(d, \mathbb{R})$ which is the group of nonsingular square matrices. Please see Section A.7.1 for more details. To derive mean shift procedure on matrix Lie groups, we define the equivalent of concepts such as distances and averages based on the geometry of the underlying space.

The most important property of the mean shift algorithm that helps us on Lie groups is *locality.* Iterations start on the data points and we can define the kernel function over the local neighborhoods of the points. We can run the mean shift algorithm on a Lie group by iteratively transforming points between the Lie group and Lie algebra.

Let $\mathbf{X}_i$, $i = 1, ..., n$, be the data points where $\mathbf{X}_i \in G$ and G is a matrix Lie group. Using the intrinsic distance (A.20), the multivariate kernel density estimate at $\mathbf{X}$ is given by

$$\hat{f}(\mathbf{X}) = \frac{c_{k,m}}{nh^m} \sum_{i=1}^{n} k\left(\left\| \frac{\log(\mathbf{X}^{-1}\mathbf{X}_i)}{h} \right\|^2 \right). \tag{2.6}$$

A similar kernel was also defined in [110] for nonparametric density estimation.

The mean shift vector is equal to the mean of the points weighted by the gradient of the density estimator centered on the previous estimate. In several applications the Lie algebra is used for computing intrinsic means on Lie groups [48, 63]. Here we follow a similar approach.

Let $\mathbf{X}$ be the current location estimation. The group operation (A.15) maps the neighborhood of $\mathbf{X}$ to the neighborhood of $\mathbf{I}$ and the tangent space at $\mathbf{X}$, $T_{\mathbf{X}}G$, to the Lie algebra $\mathfrak{g}$. Using (A.15) and the logarithm map (A.19) a first order approximation to the mean shift vector is given on the Lie algebra

$$\mathbf{m}_h(\mathbf{X}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right)}. \tag{2.7}$$

where

$$\mathbf{x}_i = \log(\mathbf{X}^{-1}\mathbf{X}_i). \tag{2.8}$$

The approximation error can be expressed in terms of the higher order terms in BCH formula (A.16). The error is minimal around $\mathbf{I}$ and the mapping (A.15) assures that the error is minimized. Moreover the point $\mathbf{X}$ is mapped to $\mathbf{0}$ via (2.8), therefore we do not have the second term of (2.5). The mean shift vector is on the Lie algebra. We transform this vector to the Lie group and update the location of $\mathbf{X}$ as

$$\bar{\mathbf{X}} = \mathbf{X}\exp(\mathbf{m}_h(\mathbf{X})). \tag{2.9}$$

Starting at a data point and iteratively updating the location with the mean shift vector, the procedure reaches a local mode of the distribution. The mean shift algorithm on Lie groups is given in Figure 2.3.

The described mean shift algorithm is applicable to any matrix Lie group. However it is important to know the convergence of the logarithm series (A.19). The exponential mapping is a homeomorphism around $\mathbf{I}$. It can be shown from the series (A.19) that the logarithm operator is convergent for $\mathbf{X}$, if

$$\|\mathbf{X} - \mathbf{I}\| < 1. \tag{2.10}$$

```
Input: Data points $\mathbf{X}_i, i = 1 \dots n$, $\mathbf{X}_i \in G$, Initial location $\mathbf{X}$
  • Repeat
      – Repeat for $i = 1...n$
          * $\mathbf{x}_i = \log(\mathbf{X}^{-1}\mathbf{X}_i)$
      – $\mathbf{m}_h(\mathbf{X}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\|\frac{\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}_i}{h}\|^2\right)}$
      – $\mathbf{X} = \mathbf{X}\exp(\mathbf{m}_h(\mathbf{X}))$
  • Until $\mathbf{m}_h(\mathbf{X}) < \varepsilon$
  • Store $\mathbf{X}$ as a local mode
```

Figure 2.3: Mean shift on matrix Lie groups.

Under mapping (A.15), $\mathbf{X}$ is transformed to $\mathbf{I}$ and the logarithm is well defined for points in a neighborhood of $\mathbf{X}$. This is good enough for us, because distant points from the current location have almost zero weights in kernel density estimation. We can simply ignore the points for which the logarithm operator does not converge. A more detailed discussion about computation of exponential and logarithm operators can be found in [3].

For 3D rigid motion estimation problem we are interested in the special orthogonal $\mathbf{SO}(3)$ and the special Euclidean $\mathbf{SO}(3)$ groups. For these parameter spaces the logarithm and exponential maps are always well defined and can be computed analytically. See Section A.7.1 for the geometry of the spaces.

2.4 Multiple Rigid Motion Estimation

We propose a very general approach to multiple rigid motion estimation. We do not make any assumption on the number of motion groups and the data might be corrupted with outliers. We estimate all the motion parameters simultaneously and do not require prior specification of the number of motion groups.

Let the two set of matched 3D measurements be $\mathbf{U} = \{\mathbf{u}_j\}_{j=1...N}$ and $\mathbf{V} = \{\mathbf{v}_j\}_{j=1...N}$. Let $\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1...p}$ be the p rigid motions. The motion parameters $\mathbf{R}_i$ is a 3×3 rotation matrix and the translation vector $\mathbf{t}_i$ is in $\mathbb{R}^3$. If the points $\mathbf{v}$ and $\mathbf{u}$ are in correspondence, the motion equation can be written as

$$\mathbf{v} = \mathbf{R}\mathbf{u} + \mathbf{t} + \boldsymbol{\eta} \tag{2.11}$$

where $\{\mathbf{R}, \mathbf{t}\}$ is one of the p motions and $\boldsymbol{\eta}$ is the measurement noise, otherwise the point correspondence is an outlier.

Our motion estimation algorithm is based on sampling and mode finding on the sampled distribution. We start sampling elemental subsets from the point correspondences and estimate the motion parameters $\mathbf{R}_i$, $\mathbf{t}_i$. This process is repeated n times and at the end of the sampling step we have a distribution of rigid motions $\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1\ldots n}$ or equivalently in $\mathbf{SE}(3)$ $\{\mathbf{M}_i\}_{i=1\ldots n}$. For rigid motions, three points are enough to estimate the motion parameters. Although we use SVD [4] algorithm to estimate motion parameters from elemental subsets, this is not a necessity and can be replaced with any rigid motion estimation method mentioned in Section 2.1. For instance, if data is corrupted with heteroscedastic noise, it would be better to use HEIV method [104].

Assume that there exist p motions and no outliers. If the same number of point correspondences belong to each motion, the probability of sampling three points from the same motion group is p^{-3}. We increase the chance of selecting points from the same motion group by adding a validation step to sampling mechanism. After sampling three points and estimating the motion $\mathbf{R}_i$ and $\mathbf{t}_i$, we select a few random points from the data set and check whether the selected points agree with the estimated motion. If any of the points agree $\|\mathbf{v}_j - \mathbf{R}_i\mathbf{u}_j - \mathbf{t}_i\| < val$ where val is the validation threshold, we keep the estimated motion, otherwise we ignore it and continue sampling. Although this step is not a necessity for our method, it reduces the computational cost of the algorithm. Sampling more elemental subsets is also a solution to sampling correct points, but during mode finding this increases the running time of the algorithm.

The next step is to find the modes of the sampled distribution. Our argument is, there should be p significant modes on the sampled distribution and these modes correspond to the motion parameters. For illustration purposes, we generated $N = 160$ 3D point correspondences from four rigid motions. The points are corrupted with

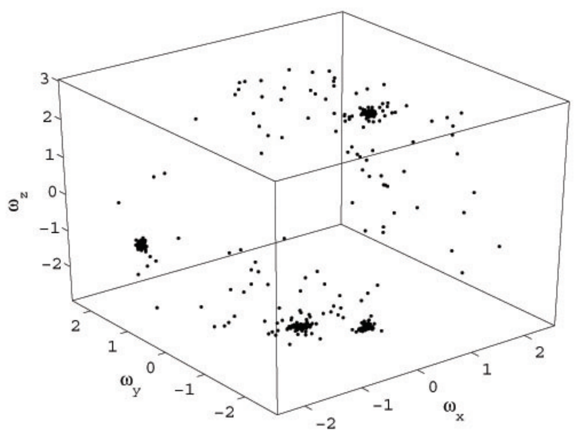


Figure 2.4: Sampled rotations mapped to Lie algebra.

zero mean unit standard deviation noise. We estimated $n = 500$ rotation matrices by sampling with validation threshold $val = 5$. Figure 2.4 shows the generated rotation distribution mapped to the Lie algebra. In this space, we clearly see the four significant modes. The outliers in the motion distribution are a direct consequence of sampling process. If all of the sampled points are not from the same motion group, the generated motion estimation is an outlier. The validation step decreases the chance of generating outliers.

In the previous section, we explained how the mean shift algorithm can be used for mode finding on Lie groups. Let $\{\mathbf{M}_i\}_{i=1...n}$ be the generated motion distribution by sampling where

$$\mathbf{M}_i = \begin{pmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}^T & 1 \end{pmatrix} \quad i = 1...n. \tag{2.12}$$

The scale of the translations $\mathbf{t}_i$ is usually much larger than the rotations $\mathbf{R}_i$. If we scale the real world coordinates and perform rigid motion estimation, we end up with the same rotations but scaled versions of translations. Using this fact, we can scale the translations. We perform zero mean, unit standard deviation normalization on the estimated translations. Finally, using mean shift we find the modes of the distribution. The number of estimated modes becomes the number of motions and the modes correspond to the motion parameters. Note that, at the end of mean shift iterations each point converges to a local mode of the distribution. In the sparse regions of the space a few points converge to a local mode. Looking at the probability densities

```
Input: Two sets of matched 3D points $\mathbf{U} = \{\mathbf{u}_j\}_{j=1...N}$ and $\mathbf{V} = \{\mathbf{v}_j\}_{j=1...N}$
  • Repeat for $i = 1...n$
      – Sample 3 corresponding points from data sets and estimate motion parameters $\mathbf{M}_i$
  • Normalize translation estimations to zero mean, unit standard deviation
  • Find modes of rigid motion distribution $\{\mathbf{M}_i\}_{i=1...n}$ via mean shift. Let $\{\hat{\mathbf{M}}_k\}_{k=1...r}$ be the detected modes
  • Report number of motions $r$
  • Renormalize translations to original scale
  • Report motion parameters $\{\hat{\mathbf{M}}_k\}_{k=1...r}$
```

Figure 2.5: Multiple rigid motion estimation.

and number of points in the basins of attraction at the local modes we eliminate the small modes. It is observed that there is a big gap in the probability densities and number of converged points between the modes generated by real motions and modes generated by random combinations of points. Therefore we easily remove the small modes. The details are explained in Section 2.5. The multiple rigid motion estimation algorithm is given in Figure 2.5.

2.5 Experimental Results

We present the results for three very challenging experiments conducted on synthetic and image data. Throughout the experiments we keep the parameters of the motion estimation algorithm fixed. For kernel density estimation (2.6), we use normal kernel profile

$$k_N(x) = e^{-\frac{1}{2}x}. \tag{2.13}$$

The bandwidth of the mean shift algorithm is selected as $h = 0.1$, and for each experiment we sampled $n = 500$ motions via elemental subsets. The sampling validation threshold, *val*, is selected as $\frac{1}{20}$ of the average motion of the points between two frames. In all our experiments mean shift algorithm correctly detected number of motions $(r = p)$ and estimated motion parameters were very accurate.

2.5.1 Error Metrics

To evaluate our results we start with defining some error metrics on the rigid motion group. The first metric is the expected error ϵ_E which is derived on $\mathbb{R}^3$ [24, p.174]. Let $\{\mathbf{R}_T, \mathbf{t}_T\}$ be the true motion and $\{\mathbf{R}_E, \mathbf{t}_E\}$ be the estimated motion. The expected error can be measured by

$$\epsilon_E = \int_{\mathbb{R}^3} \|\mathbf{R}_T\mathbf{x} + \mathbf{t}_T - \mathbf{R}_E\mathbf{x} - \mathbf{t}_E\| \rho(\mathbf{x}) d\mathbf{x} \tag{2.14}$$

where $\rho(\mathbf{x})$ is ideally the mass density of the object effected by the motion. Instead, we assume $\rho(\mathbf{x})$ is uniformly distributed and replace the integral with a finite summation

$$\epsilon_E = \frac{1}{c} \sum_{i=1}^{c} \|\mathbf{R}_T\mathbf{x}_i + \mathbf{t}_T - \mathbf{R}_E\mathbf{x}_i - \mathbf{t}_E\| \tag{2.15}$$

where we generate c random points $\{\mathbf{x}_i\}_{i=1..c}$. The metric is equivalent to expectation of error for a point on the rigid body. Although it might overestimate the error, during our test we generate $c = 100$ random points in the maximum range of the points in the original set and compute the expected error (2.15) for each motion on this set.

The other error metrics are computed independently on rotation and translation estimation. The second error metric is the rotation error. We measure the rotation error based on matrix norms [24, p.143]

$$\epsilon_R = \|\mathbf{R}_T^{-1}\mathbf{R}_E - \mathbf{I}\|_F \tag{2.16}$$

where $\|.\|_F$ denotes the Frobenius norm of a matrix.

The last error metric is the translation error which can be directly measured with the Euclidean distance

$$\epsilon_T = \|\mathbf{t}_T - \mathbf{t}_E\|_2. \tag{2.17}$$

2.5.2 Synthetic Data

In the first experiment, we generated 3D points in $[0, 100]$ range and transformed these points according to four different rigid motions. Each motion acts on 25 points. We add zero mean, unit standard deviation Gaussian noise to each coordinate of the points in the original and the transformed set. Moreover, we add 100 outlier points to the

	Multiple Motion			SVD		
	ϵ_E	ϵ_R	ϵ_T	ϵ_E	ϵ_R	ϵ_T
$\mathbf{M}_1$	0.5166	0.0095	0.9330	0.7885	0.0163	1.0393
$\mathbf{M}_2$	0.5559	0.0110	0.6185	0.6350	0.0202	0.6198
$\mathbf{M}_3$	0.7875	0.0138	0.8344	0.6759	0.0123	0.4823
$\mathbf{M}_4$	0.6785	0.0248	0.5490	0.6557	0.0216	0.3327

Table 2.1: Estimation errors on synthetic data. The SVD algorithm is performed separately for each motion by manually removing all points from other motions and outliers.

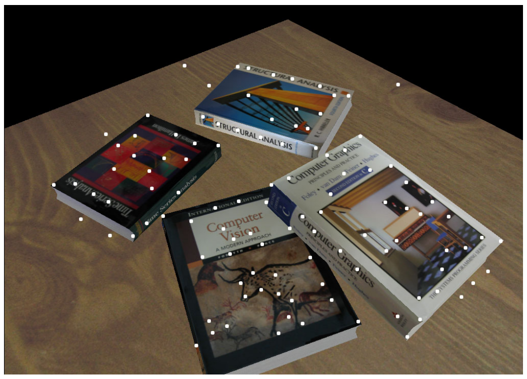
original and transformed set by generating random points. As a result, we have 100 3D point correspondences from four motions and 100 mismatched points. As can be imagined this is a very challenging situation. According to each motion group only 1/8 of the points are inliers and the noise is high. The mean shift algorithm found four motions and the errors associated with each motion are shown in Table 2.1.

We compare our results with the SVD algorithm performed on each motion *separately* and *outliers removed*. Note that, the SVD algorithm can not estimate multiple motions or if there exists outliers. The results show that, although we estimate multiple motions in the presence of significant amount of outliers, our algorithm performs as good as or better than the SVD algorithm performed separately on each motion with the outliers removed.

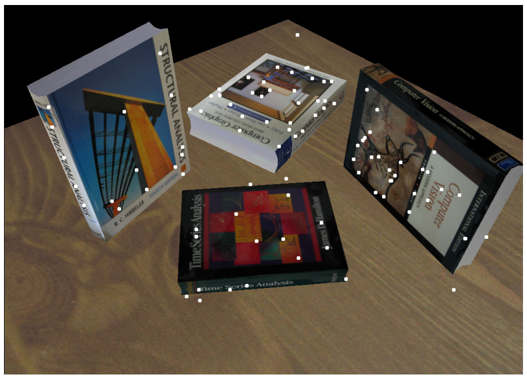
2.5.3 3D Computer Generated Image Data

The second experiment is performed on computer generated 3D image data. To evaluate the performance of our algorithm we need to know the exact transformations applied on the bodies, therefore we created a 3D scene. We found the point correspondences by finding salient features on the original image and tracking these features in the transformed image. Corners on the original image are found by a Harris corner detector [68] and corresponding points in the transformed image are found via the point matching algorithm described in [61]. Using depth buffer and camera matrices the 3D coordinates of each pixel are recovered.

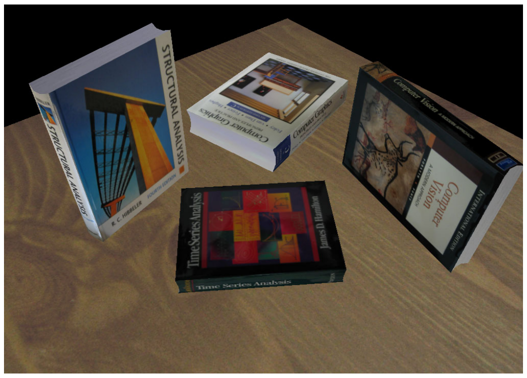
A total of $N = 112$ points are detected by the corner detection algorithm. The detected corner points are shown in Figure 2.6a and corresponding points are shown in



(a)



(b)



(c)

Figure 2.6: 3D image data. (a) Original scene. 112 points are detected via corner detection algorithm. (b) Transformed scene. Corresponding points are found via point matching algorithm. (c) Reconstructed scene from (a) with the estimated motion parameters. It is very hard to see the difference from (b).

Figure 2.6b by white dots. As seen in the figures, the point matching algorithm failed for several points and only around half of the points were correctly matched. Moreover, for two bodies there exist only around 10 point matches. The mean shift algorithm reported four motions. We compared our results with the true motions and errors are given in Table 2.2. Correspondences are generated via point matching algorithm and we do not know the true points. Therefore, SVD algorithm could not be used for comparison. The errors indicate that results are close to perfect. Figure 2.6c shows the reconstructed scene form the original scene (Figure 2.6a) with the estimated motion parameters. It is very hard to detect the difference from the original transformed image (Figure 2.6b).

	ϵ_E	ϵ_R	ϵ_T
$\mathbf{M}_1$	0.4653	0.0425	0.2454
$\mathbf{M}_2$	0.1385	0.0071	0.1293
$\mathbf{M}_3$	0.3350	0.0180	0.2992
$\mathbf{M}_4$	0.5188	0.0391	0.3508

Table 2.2: Estimation errors on 3D image data.

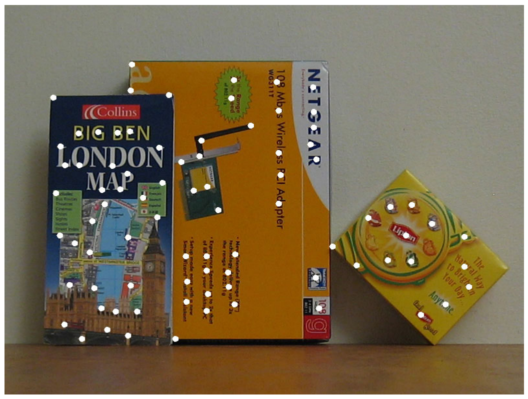
	Experiment-1		Experiment-2		Experiment-3	
	Points	Pdf	Points	Pdf	Points	Pdf
$\mathbf{M}_1$	77	0.1051	79	0.0901	196	0.3394
$\mathbf{M}_2$	72	0.0950	23	0.0401	184	0.2951
$\mathbf{M}_3$	74	0.0844	34	0.0360	55	0.0921
$\mathbf{M}_4$	50	0.0620	22	0.0336	**4**	**0.0087**
$\mathbf{M}_5$	**2**	**0.0040**	**4**	**0.0161**	6	0.0081
$\vdots$	$\vdots$					

Table 2.3: Number of points in the basins of attraction and the probability densities at the detected modes in the three experiments. The local modes are sorted according to pdfs. The modes corresponding to random motions are detected using the number of points and pdfs. The first random modes are shown in bold. In all of the experiments the number of motions is very clear.

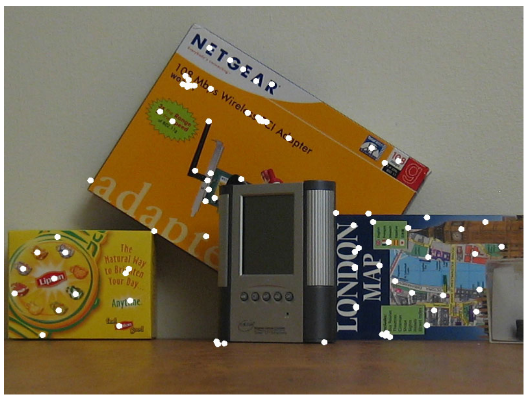
2.5.4 2D Real Image Data

The third experiment is conducted on 2D images in a real scene. In this experiment we estimate multiple 2D rigid motions in the presence of occlusions. There are three rigid bodies in the original scene and in the transformed image two of the bodies are occluded by some other objects. The corner detection algorithm found $N = 83$ points (Figure 2.7a) and matching points are again found via [61] (Figure 2.7b). Due to occlusions and errors in the point matching process most of the point correspondences are outliers. The mean shift algorithm reported three motions. We manually segmented the boundaries of the bodies in the original image by marking the four corners of the bodies and transformed the boundaries according to the estimated motions. The result is presented in Figure 2.7d. Note that in this simple 2D case, the Lie algebra parametrization is equivalent to angle-translation parametrization of 2D rigid motion.

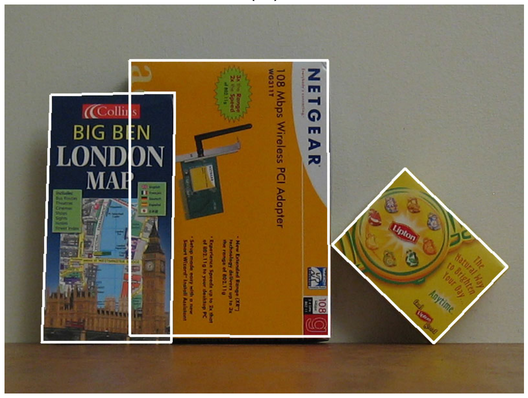
In Table 2.3, we show the number of points in the basins of attraction and probability densities for each of the local modes detected by mean shift algorithm. The density at the mode is estimated via (2.6). Looking at the results it is very clear which clusters



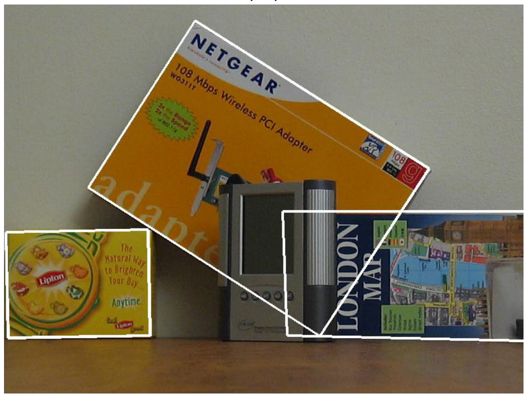
(a)



(b)



(c)



(d)

Figure 2.7: 2D image data. (a) Original scene. 83 points are detected via corner detection algorithm. (b) Transformed scene. Corresponding points are found via point matching algorithm. (c) The boundaries of the bodies. (d) Transformed boundaries with the estimated motion parameters. The estimation is almost perfect.

are due to random combinations of the points and which clusters are generated due to actual motions. We remove the modes having less than 10 points in the basins of attraction or having less than $\frac{1}{10}$ probability of the most significant mode.

2.5.5 Computational Requirement

The total processing time of sampling 500 motions and running mean shift on $\mathbf{SE}(3)$ is less than one minute on a Pentium IV 3.2Ghz processor with a C++ implementation. The exponential and logarithm operators are implemented using the matrix series [3]. The computational time can be significantly reduced using the analytic forms described in Section A.7.1. The 2D and 3D image data used in the experiments can be downloaded from `www.caip.rutgers.edu/riul/robust.html`.

Chapter 3

Region Covariance Descriptors

3.1 Introduction

Feature selection is one of the most important steps for computer vision problems. Good features should be discriminative, robust, easy to compute and efficient algorithms are needed for a variety of tasks such as recognition and tracking.

The raw pixel values of several image statistics such as color, gradient and filter responses are the simplest choice for image features, and were used for many years in computer vision, e.g., [17, 103, 132]. However, these features are not robust in the presence of illumination changes and nonrigid motion, and efficient matching algorithms are limited by the high dimensional representation. Lower dimensional projections were also used for classification [152] and tracking [12].

A natural extension of raw pixel values are via histograms where a region is represented with its nonparametric estimation of joint distribution. Following [29], histograms were widely used for nonrigid object tracking. In a recent study [123], fast histogram construction methods were explored to find a global match. Besides tracking, histograms were also used for texture representation [98, 162], matching [61] and other problems in the field of computer vision. However, the joint representation of several different features through histograms is exponential with the number features.

Haar wavelet based descriptors [121] are a set of basis functions which encode the intensity differences between two regions. Combined with cascaded AdaBoost classifier, superior performances were reported for face detection problem, but the algorithm requires long training time to learn an object class. In [100] scale space extremas are detected for keypoint localization and arrays of orientation histograms were used as keypoint descriptors. The descriptors are very effective in matching local neighborhoods but do not have global context information.

We describe a novel region descriptor and apply it to three challenging problems, matching, texture classification and tracking [154]. The covariance of d-features, e.g., the three-dimensional color vector, the norm of first and second derivatives of intensity with respect to x and y, etc., characterizes a region of interest. We describe a fast method for computation of covariances based on integral images. The idea presented here is more general than the image sum or histogram, which were already described in the literature, and with a series of integral images the covariances are obtained by a few arithmetic operations.

Covariance matrices do not form a vector space but lie on a smooth manifold, therefore the Euclidean distance is not an appropriate metric for the space. We use a distance involving generalized eigenvalues of matrices which also follows from an affine invariant metric on the space of symmetric positive definite matrices (nonsingular covariance matrices). Feature matching is a simple nearest neighbor search under the distance metric and performed extremely rapidly using the integral representation.

3.2 Covariance as a Region Descriptor

3.2.1 Covariance Descriptors

Let I be a one dimensional intensity or three dimensional color image. The method also generalizes to other type of images, e.g., infrared. Let F be the $W \times H \times d$ dimensional feature image extracted from I

$$F(x, y) = \phi(I, x, y) \tag{3.1}$$

where the function ϕ can be any mapping such as intensity, color, gradients, filter responses, etc. For a given rectangular region $R \subset F$, let $\{\mathbf{z}_k\}_{k=1..S}$ be the d-dimensional feature points inside R. We represent the region R with the $d \times d$ covariance matrix of the feature points

$$\mathbf{C}_R = \frac{1}{S-1} \sum_{k=1}^{n} (\mathbf{z}_k - \boldsymbol{\mu})(\mathbf{z}_k - \boldsymbol{\mu})^T \tag{3.2}$$

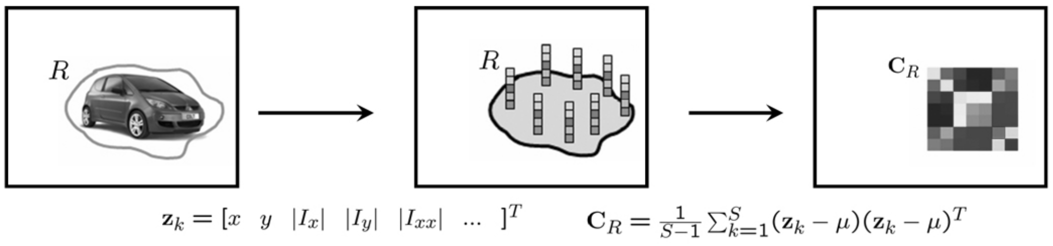

R
R
$\mathbf{C}_R$
$\mathbf{z}_k = [x \quad y \quad |I_x| \quad |I_y| \quad |I_{xx}| \quad \ldots \quad]^T$
$\mathbf{C}_R = \frac{1}{S-1}\sum_{k=1}^{S}(\mathbf{z}_k - \mu)(\mathbf{z}_k - \mu)^T$

Figure 3.1: Covariance descriptor. The d-dimensional feature image F is constructed from input image I through mapping Φ. The region R is represented with the covariance matrix, $\mathbf{C}_R$, of the features $\{\mathbf{z}_k\}_{k=1..S}$.

where $\boldsymbol{\mu}$ is the mean of the points. In Figure 3.1, we delineate the construction of covariance descriptors.

There are several advantages of using covariance matrices as region descriptors. A single covariance matrix extracted from a region is usually enough to match the region in different views and poses. In fact we assume that the covariance of a distribution is enough to discriminate it from other distributions. If two distributions only vary with their mean, our matching result produces perfect match but in real examples these cases almost never occur.

The covariance matrix proposes a natural way of fusing multiple features which might be correlated. The diagonal entries of the covariance matrix represent the variance of each feature and the nondiagonal entries represent the correlations. The noise corrupting individual samples are largely filtered out with an average filter during covariance computation.

The covariance matrices are low-dimensional compared to other region descriptors and due to symmetry $\mathbf{C}_R$ has only $(d^2 + d)/2$ different values. Whereas if we represent the same region with raw values we need $n \times d$ dimensions, and if we use joint feature histograms we need b^d dimensions, where b is the number of histogram bins used for each feature.

Given a region R, its covariance $\mathbf{C}_R$ does not have any information regarding the ordering and the number of points. This implies a certain scale and rotation invariance over the regions in different images. Nevertheless, if information regarding the orientation of the points are represented, such as the norm of gradient with respect to x and

y, the covariance descriptor is no longer rotationally invariant. The same argument is also correct for scale and illumination. Rotation and illumination dependent statistics are important for recognition/classification purposes and we use them in Sections 3.3 and 3.4.

3.2.2 Distance Calculation on Covariance Matrices

The covariance matrices do not form a vector space. For example, the space is not closed under multiplication with negative scalers. Most of the common machine learning algorithms assume that the data points form a vector space therefore they are not suitable for our descriptors. The nearest neighbor algorithm which will be used in the following sections, only requires a way of computing distances between feature points. We use the distance measure proposed in [52] to measure the dissimilarity of two covariance matrices

$$\rho(\mathbf{C}_1, \mathbf{C}_2) = \sqrt{\sum_{i=1}^{d} \ln^2 \lambda_i(\mathbf{C}_1, \mathbf{C}_2)} \tag{3.3}$$

where $\{\lambda_i(\mathbf{C}_1, \mathbf{C}_2)\}_{i=1...d}$ are the generalized eigenvalues of $\mathbf{C}_1$ and $\mathbf{C}_2$, computed from

$$\lambda_i \mathbf{C}_1 \mathbf{x}_i - \mathbf{C}_2 \mathbf{x}_i = 0 \qquad i = 1...d \tag{3.4}$$

and $\mathbf{x}_i \neq 0$ are the generalized eigenvectors. The distance measure ρ satisfies the metric axioms for positive definite symmetric matrices $\mathbf{C}_1$ and $\mathbf{C}_2$

1. $\rho(\mathbf{C}_1, \mathbf{C}_2) \geq 0$ and $\rho(\mathbf{C}_1, \mathbf{C}_2) = 0$ only if $\mathbf{C}_1 = \mathbf{C}_2$,
2. $\rho(\mathbf{C}_1, \mathbf{C}_2) = \rho(\mathbf{C}_2, \mathbf{C}_1)$,
3. $\rho(\mathbf{C}_1, \mathbf{C}_2) + \rho(\mathbf{C}_1, \mathbf{C}_3) \geq \rho(\mathbf{C}_2, \mathbf{C}_3)$.

The generalized eigenvalues can be computed with $\mathcal{O}(d^3)$ arithmetic operations using numerical methods and an additional d logarithm operations are required for distance computation, which is usually faster than comparing two histograms that grow exponentially with d.

The distance metric is equivalent to (A.34) which can be derived using the affine invariant metric (A.29) defined on the tangent space of symmetric positive definite matrices. See Section A.7.2 for more details.

3.2.3 Integral Images for Fast Covariance Computation

Integral images are intermediate image representations used for fast calculation of region sums [144, 165]. Each pixel of the integral image is the sum of all the pixels inside the rectangle bounded by the upper left corner of the image and the pixel of interest. For an intensity image I its integral image is defined as

$$\text{Integral Image } (x', y') = \sum_{x \leq x', y \leq y'} I(x, y). \tag{3.5}$$

Using this representation, any rectangular region sum can be computed in constant time. In [123], the integral images were extended to higher dimensions for fast calculation of region histograms. Here we follow a similar idea for fast calculation of region covariances [126].

We can write the (i, j)-th element of the covariance matrix defined in (3.2) as

$$C_R(i, j) = \frac{1}{S-1} \sum_{k=1}^{S} (z_k(i) - \mu(i))(z_k(j) - \mu(j)). \tag{3.6}$$

Expanding the mean and rearranging the terms we can write

$$C_R(i, j) = \frac{1}{S-1} \left[\sum_{k=1}^{S} z_k(i) z_k(j) - \frac{1}{S} \sum_{k=1}^{S} z_k(i) \sum_{k=1}^{S} z_k(j) \right]. \tag{3.7}$$

To find the covariance in a given rectangular region R, we have to compute the sum of each feature dimension, $z(i)_{i=1...d}$, as well as the sum of the multiplication of any two feature dimensions, $z(i)z(j)_{i,j=1...d}$. We construct $d + d^2$ integral images for each feature dimension $z(i)$ and multiplication of any two feature dimensions $z(i)z(j)$.

Let P be the $W \times H \times d$ tensor of the integral images

$$P(x', y', i) = \sum_{x \leq x', y \leq y'} F(x, y, i) \qquad i = 1...d \tag{3.8}$$

and Q be the $W \times H \times d \times d$ tensor of the second order integral images

$$Q(x', y', i, j) = \sum_{x \leq x', y \leq y'} F(x, y, i) F(x, y, j) \qquad i, j = 1...d. \tag{3.9}$$

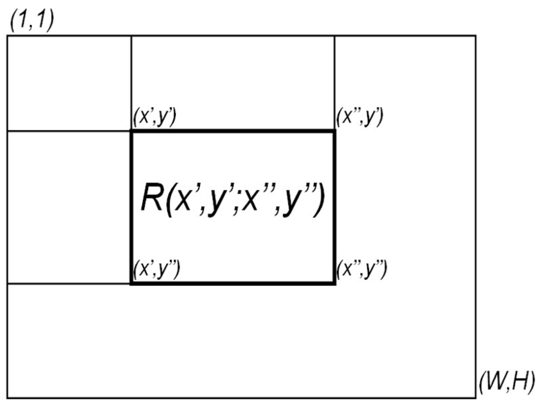


Figure 3.2: Integral Image. The rectangle $R(x', y'; x'', y'')$ is defined by its upper left (x', y') and lower right (x'', y'') corners in the image, and each point is a d dimensional vector.

In [165], it is shown that integral image can be computed in one pass over the image. In our notation, $\mathbf{p}_{x,y}$ is the d dimensional vector and $\mathbf{Q}_{x,y}$ is the $d \times d$ dimensional matrix

$$\begin{aligned} \mathbf{p}_{x,y} &= [P(x,y,1)\dots P(x,y,d)]^T \\ \mathbf{Q}_{x,y} &= \begin{pmatrix} Q(x,y,1,1) & \dots & Q(x,y,1,d) \\ & \vdots & \\ Q(x,y,d,1) & \dots & Q(x,y,d,d) \end{pmatrix}. \end{aligned} \tag{3.10}$$

Note that $\mathbf{Q}_{x,y}$ is a symmetric matrix and $d + (d^2 + d)/2$ passes over the image are enough to compute both P and Q. The computational complexity of constructing the integral images is $\mathcal{O}(d^2WH)$.

Let $R(x', y'; x'', y'')$ be the rectangular region, where (x', y') is the upper left coordinate and (x'', y'') is the lower right coordinate, as shown in Figure 3.2. The covariance of the region bounded by $(1, 1)$ and (x', y') is

$$\mathbf{C}_{R(1,1;x',y')} = \frac{1}{S-1}\left[\mathbf{Q}_{x',y'} - \frac{1}{S}\mathbf{p}_{x',y'}\mathbf{p}^T_{x',y'}\right] \tag{3.11}$$

where $S = x' \cdot y'$. Similarly, after a few rearrangements, the covariance of the region $R(x', y'; x'', y'')$ can be computed as

$$\begin{aligned} \mathbf{C}_{R(x',y';x'',y'')} = & \frac{1}{S-1}\Big[\mathbf{Q}_{x'',y''} + \mathbf{Q}_{x'-1,y'-1} - \mathbf{Q}_{x'',y'-1} - \mathbf{Q}_{x'-1,y''} \\ & -\frac{1}{S}\big(\mathbf{p}_{x'',y''} + \mathbf{p}_{x'-1,y'-1} - \mathbf{p}_{x'-1,y''} - \mathbf{p}_{x'',y'-1}\big) \\ & \big(\mathbf{p}_{x'',y''} + \mathbf{p}_{x'-1,y'-1} - \mathbf{p}_{x'-1,y''} - \mathbf{p}_{x'',y'-1}\big)^T\Big] \end{aligned} \tag{3.12}$$

where $S = (x'' - x' + 1) \cdot (y'' - y' + 1)$. Therefore, after constructing integral images the covariance of any rectangular region can be computed in $\mathcal{O}(d^2)$ time.

3.3 Region Matching

The aim of region matching is to locate a given object of interest in an arbitrary image and pose, after a possible nonrigid transformation. We use pixel locations (x,y), color (RGB) values and the norm of the first and second order derivatives of the intensities with respect to x and y. Each pixel of the image is converted to a nine-dimensional feature vector

$$F(x,y) = \left[x \;\; y \;\; R(x,y) \;\; G(x,y) \;\; B(x,y) \;\; |I_x(x,y)| \;\; |I_y(x,y)| \;\; |I_{xx}(x,y)| \;\; |I_{yy}(x,y)|\right]^T \tag{3.13}$$

where R, G, B are the RGB color values, and $I_x, I_{xx}, ..$ are intensity derivatives. The image derivatives are calculated through the filters $[-1\ 0\ 1]^T$ and $[-1\ 2\ -1]^T$, for first and second order respectively. The covariance of a region is a 9×9 matrix. Although the variance of pixel locations (x,y) is same for all the regions of the same size, they are still important since their correlation with the other features are used at the nondiagonal entries of the covariance matrix.

We represent an object with five covariance matrices of the image features computed inside the object region, as shown in Figure 3.3. Initially we compute only the covariance of the whole region, $\mathbf{C}_1$, from the source image. We search the target image for a region having similar covariance matrix and the dissimilarity is measured through (3.3). At all the locations in the target image we analyze at nine different scales (four smaller, four larger) to find matching regions. We perform a brute force search, since we can compute the covariance of an arbitrary region very quickly. Instead of scaling the target image, we just change the size of our search window. There is a 15% scaling factor between two consecutive scales. The variance of the x and y components are not the same for regions with different sizes and we normalize the rows and columns corresponding to these features. At the smallest size of the window we jump three pixels horizontally or vertically between two search locations. For larger windows we jump 15% more and

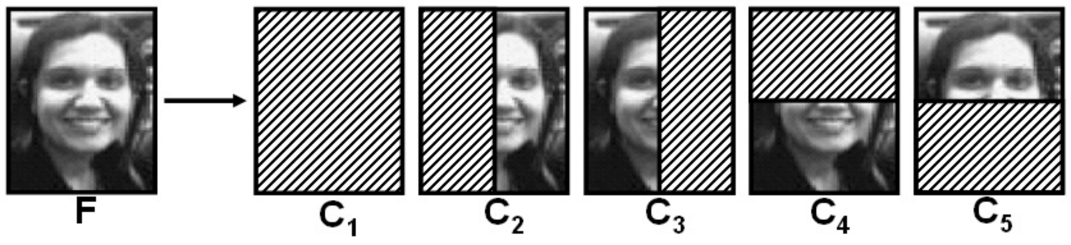


Figure 3.3: Object representation. We construct five covariance matrices from overlapping regions of an object feature image. The covariances are used as the object descriptors.

round to the next integer at each scale.

We keep the best matching 1000 locations and scales. At the second phase we repeat the search for 1000 detected locations, using the covariance matrices $\mathbf{C}_{i=1...5}$. The dissimilarity of the object model and a target region is computed

$$\rho(O, M) = \min_j \left[\sum_{i=1}^{5} \rho(\mathbf{C}_i^O, \mathbf{C}_i^T) - \rho(\mathbf{C}_j^O, \mathbf{C}_j^M) \right] \tag{3.14}$$

where $\mathbf{C}_i^O$ and $\mathbf{C}_i^M$ are the object and target covariances respectively, and we ignore the least matching region covariance of the five. This increases robustness towards possible occlusions and large illumination changes. The region with the smallest dissimilarity is selected as the matching region.

We present the matching results for a variety of examples in Figure 3.4 and compare our results with histogram features. We tested histogram features both with the RGB and HSV color spaces. With the RGB color space the results were much worse in all of the cases, therefore we did not present these results. We construct three separate 64 bin histograms for hue, saturation and value since it is not practical to construct a joint histogram. We search the target image for the same locations and sizes, and fast construction of histograms are performed through integral histograms [123]. We measure the distance between two histograms through Bhattacharyya distance [29] and sum over three color channels.

Covariance features can match all the target regions accurately whereas most of the regions found by histogram are erroneous. Even among the correctly detected regions with both methods we see that covariance features better localize the target.

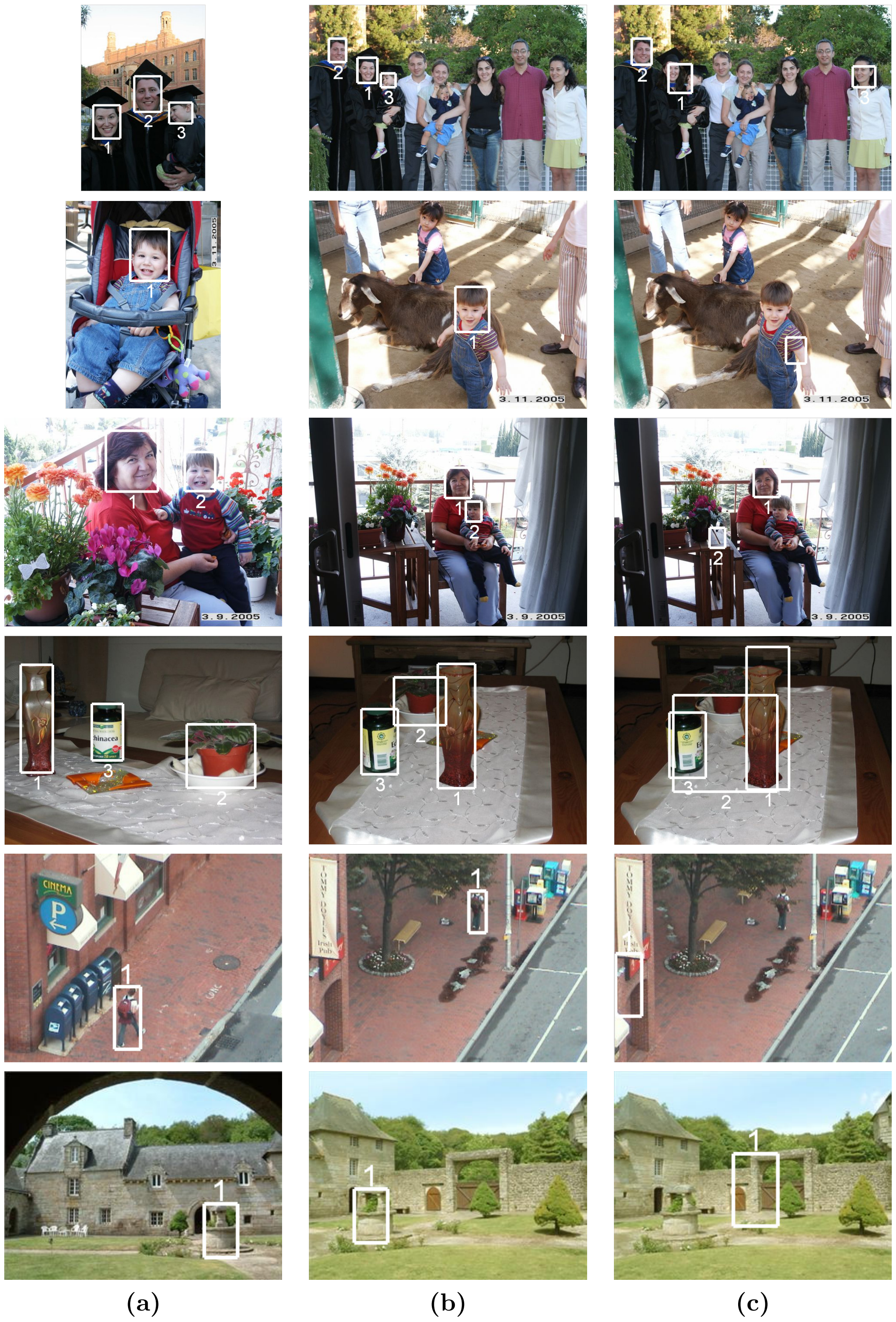


Figure 3.4: Object detection. (a) Input regions. (b) Regions found via covariance features. (c) Regions found via histogram features.

The examples are challenging since there are large scale, orientation and illumination changes, and some of the targets are occluded and have nonrigid motion. Almost perfect results indicate the robustness of the proposed approach. We also conclude that the covariances are very discriminative since they can match the correct target in the presence of similar objects, as seen in the face matching examples.

Covariance features are faster than the integral histograms since the dimensionality of the space is smaller. The search time of an object in a color image with size 320×240 is 6.5 seconds with a MATLAB 7 implementation. The performance can be improved by a factor of 20-30 with a C++ implementation which would yield to near real time performance.

3.4 Texture Classification

Currently, the most successful methods for texture classification are through textons which are cluster centers in a feature space derived from the input. The feature space is built from the output of a filter bank applied at every pixel and the methods differ only in the employed filter bank.

- **LM**: A combination of 48 anisotropic and isotropic filters were used by Leung and Malik [98]. The feature space is 48 dimensional.
- **S**: A set of 13 circular symmetric filters was used by Schmid [138]. The feature space is 13 dimensional.
- **M4**, **M8**: Both representations were proposed by Varma and Zissermann [162]. Original filters include both rotationally symmetric and oriented filters but only maximum response oriented filters are included to feature vector. The feature space is 4 and 8 dimensional respectively.

To find the textons, usually the k-means clustering algorithm is used, although it was shown that it might not be the best choice [62]. The most significant textons are aggregated into the texton library and the texton histograms are used as texture representation. The χ^2 distance [98] is used to measure the similarity of two histograms and

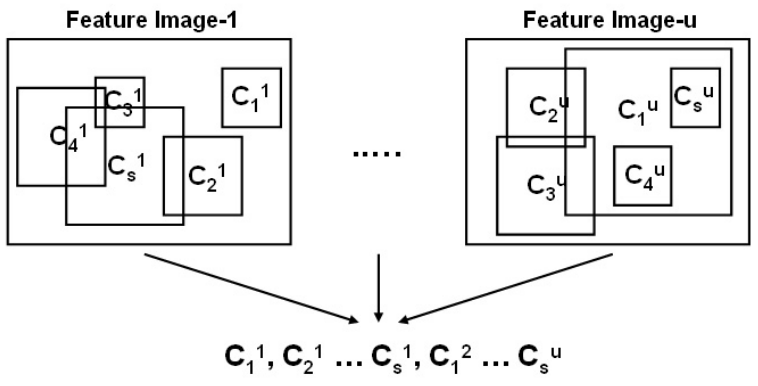


Figure 3.5: Texture representation. There are u images for each texture class and we sample s regions from each image and compute covariance matrices $\mathbf{C}$.

the training image with the smallest distance from the test image determines the class of the latter. The process is computationally expensive since the images are convolved with large filter banks and in most cases requires clustering in high dimensional space.

3.4.1 Random Covariances for Texture Classification

We present a new approach to texture classification problem without using textons. We start with extracting several features from each pixel. For texture classification problem we use image intensities and norms of first and second order derivatives of intensities in both x and y direction. Each pixel is mapped to a $d = 5$ dimensional feature space

$$F(x,y) = \Big[I(x,y) \quad |I_x(x,y)| \quad |I_y(x,y)| \quad |I_{xx}(x,y)| \quad |I_{yy}(x,y)| \Big]^T . \tag{3.15}$$

We sample s random square regions from each image with random sizes between 16×16 and 128×128. Using integral images we compute the covariance matrix of each region. Each texture image is then represented with s covariance matrices and we have u training texture images from each texture class, a total of $s \cdot u$ covariance matrices. Texture representation process is illustrated in Figure 3.5. We repeat the process for the c texture classes and construct the representation for each texture class in the same way.

Given a test image, we again extract s covariance matrices from randomly selected regions. For each covariance matrix we measure the distance (3.3) from all the matrices of the training set and the label is predicted according to the majority voting among

	M4	**M8**	**S**	**LM**	**Random Covariance**
Performance	85.71	94.64	93.30	97.32	97.77

Table 3.1: Classification results for the Brodatz database.

the k nearest ones (kNN algorithm). This classifier performs as a weak classifier and the class of the texture is determined according to the maximum votes among the s weak classifiers.

3.4.2 Texture Classification Experiments

We perform our tests on the Brodatz texture database which consists of 112 textures. Because of the nonhomogeneous textures inside the database, classification is a challenging task. We duplicate the test environment of [62]. Each 640×640 texture image is divided into four 320×320 subimages and half of the images are used for training and half for testing.

We compare our results with the results reported in [62] in Table 3.1. Here we present the results for k-means based clustering algorithm. The texture representation through texton histograms has 560 bins. The results vary from 85.71% to 97.32% depending on the filter bank used.

In our tests we sample $s = 100$ random covariances from each image, both for testing and training, and we used $k = 5$ for the kNN algorithm. For $d = 5$ dimensional features, the covariance matrix is 5×5 and has only 15 different values compared to 560 bins before. Our result, 97.77%, is better than all of the previous results and faster. Only 5 images out of 224 is misclassified which is close to the upper limit of the problem. We show the misclassified images in Figure 3.6 and the misclassifications are usually in nonhomogeneous textures.

To make the method rotationally invariant, we used only three rotationally invariant features: intensity and the magnitude of the gradient and Laplacian. The covariance matrices are 3×3 and have only 6 different values. Even with this very simple features the classification performance is 94.20%, which is as good as or even better than other rotationally invariant methods (**M4**, **M8**, **S**) listed in Table 3.1. Due to random

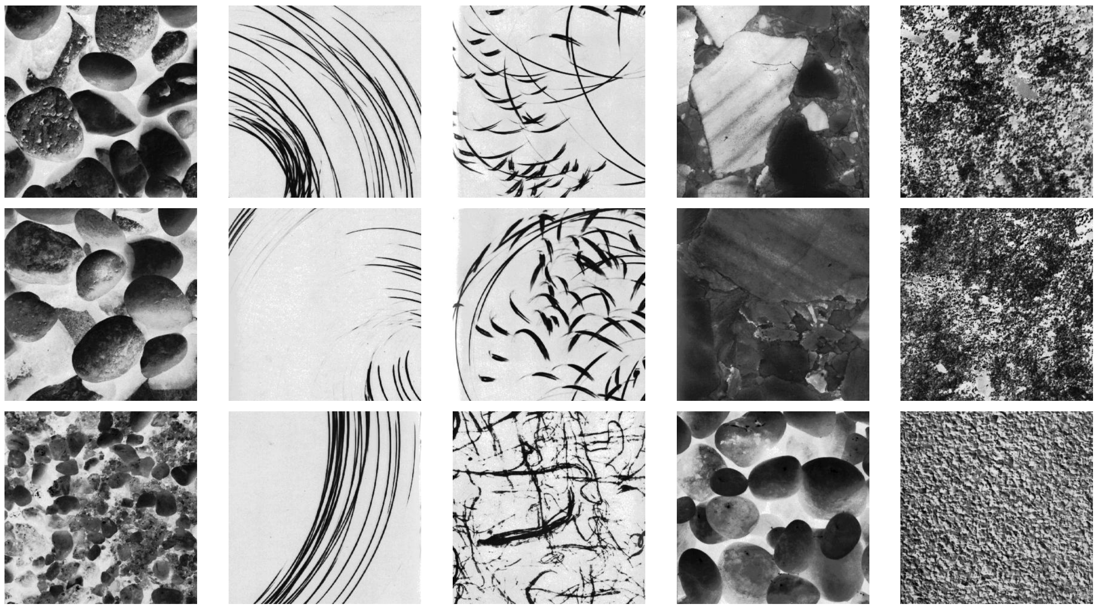


Figure 3.6: Misclassified textures. (First Row) Test examples. The misclassifications are usually in nonhomogeneous textures. (Second Row) Samples from the same class. (Third Row) Samples from the predicted texture class.

sized window selection our method is scale invariant. Although the approach is not completely illumination invariant, it is more robust than using features (intensity and gradients) directly. The variances of intensity and gradients inside regions change less than intensity and gradients themselves in illumination variations.

In the *second experiment* we compare the covariance features with other possible choices. We run the proposed texture classification algorithm with the raw intensity values and histograms extracted from random regions.

For raw intensities we normalize each random region to 16×16 square region and use Euclidean distance to compute distances for kNN classification, which is similar to [103]. The feature space is 256 dimensional. The raw intensity values are very noisy therefore only in this case we sample $s = 500$ regions from each image.

We perform two tests using histogram features: intensity only, and intensity and norms of first and second order derivatives together. In both cases the dissimilarity is measured with Bhattacharyya distance [29]. We use 256 bins for intensity only and $5 \cdot 64 = 320$ bins for intensity and norm of derivatives together. It is not practical to construct the joint intensity and norm of derivatives histograms, due to computational

	Raw Int.	**Int. Hist.**	**Int./Der. Hist.**	**Covariance**
Performance	26.79	83.35	96.88	97.77

Table 3.2: Classification results for different features.

and memory requirement.

We sample $s = 100$ regions from each texture image. The results are shown in Table 3.2. The only result close to covariance is the 320 dimensional intensity and derivative histograms together. This is not surprising because our covariance features are the covariances of the joint distribution of the intensity and derivatives. But with covariance features we achieve a better performance in a much faster way.

3.5 Tracking

Finding the correspondences of the previously detected objects in the current frame, tracking, is an essential component of several vision applications. Still, robust and accurate tracking of a deforming, non-rigid and fast moving object without getting restricted to particular model assumptions presents a major challenge.

Here we briefly describe the conventional tracking methods and their latent shortcomings. Mean shift tracker [29] is a nonparametric density gradient estimator to find the image window that is most similar to the object's color histogram in the current frame. It iteratively carries out a kernel based search starting at the previous location of the object. Even though there are variants [125] to improve its localization by using additional modalities, the original method requires the object kernels in the consecutive frames to have a certain overlap. The success of the mean-shift highly depends on the discriminating power of the histograms that are considered as the objects' probability density function.

Tracking can be considered as estimation of the state given all the measurements up to that moment, or equivalently constructing the probability density function of object location. A common approach is to employ predictive filtering and use the statistics of object's color and location in the distance computation while updating the object model by constant weights [172]. When the measurement noise are assumed to be Gaussian, the optimal solution is provided by the Kalman filter [16]. When the state space is discrete and consists of a finite number of states, Markovian filters can be applied for tracking. The most general class of filters is represented by particle filters, which are based on Monte Carlo integration methods. The current density of the state (which can be location, size, speed, boundary [80], etc.) is represented by a set of random samples with associated weights and the new density is computed based on these samples and weights. Particle filtering is a popular tracking method [15, 20, 179]. However, it is based on random sampling that becomes a problematic issue due to sample degeneracy and impoverishment, especially for higher dimensional representations.

Tracking can also be considered as a classification problem and a classifier can be trained to distinguish the object from the background [5]. This is done by constructing a feature vector for every pixel in the reference image and training a classifier to separate pixels that belong to the object from pixels that belong to the background. As in the mean-shift, an object can be tracked only if its motion is small. One obvious drawback of the local search methods is that they tend to stuck into the local optimum.

3.5.1 Covariance Tracker

Covariance tracker [127, 124] is a generalization of the region matching framework presented in Section 3.3 for tracking problem. The target object is initialized manually. At each frame, we construct a feature image using mapping (3.1). The object model is given by the covariance descriptor of the object region which is computed at the initial frame. The covariance descriptor allows us to fuse multiple modalities such as color and infrared measurements, and captures both statistical and spatial properties. The location of the target in the current frame is given by the region having minimum covariance distance from the object model. Unlike the local approaches, we find a global minimum of the distance function and the search is performed extremely rapidly using the integral representation (Section 3.2.3).

Since non-rigid and moving objects undergo shape, size, and appearance transformations in time, it is necessary to adapt to these variations. Let $\{R_t\}_{t=1...T}$ and $\{S_t\}_{t=1...T}$

be the estimated locations and their sizes during tracking. In case all the corresponding feature measurements $\{\mathbf{z}_{t,k}\}_{k=1\ldots S_t}$, $\mathbf{z}_{t,k} \in R_t$ are stored, an aggregated covariance matrix at time T can be obtained by

$$\bar{\mathbf{C}} = \frac{1}{\sum_{t=1}^{T} S_t - 1} \sum_{t=1}^{T} \sum_{k=1}^{S_t} (\mathbf{z}_{t,k} - \bar{\boldsymbol{\mu}}_T)(\mathbf{z}_{t,k} - \bar{\boldsymbol{\mu}}_T)^T \tag{3.16}$$

where $\bar{\boldsymbol{\mu}}$ is the aggregated mean computed over all regions $\{R_t\}_{t=1\ldots T}$

$$\bar{\boldsymbol{\mu}} = \frac{1}{\sum_{t=1}^{T} S_t} \sum_{t=1}^{T} \sum_{k=1}^{S_t} \mathbf{z}_{t,k}. \tag{3.17}$$

Although this formulation is arguably straightforward, it assumes that the larger windows are more influential and there is no temporal weighting for more recent observations. Besides, it is computationally expensive, $\mathcal{O}(\bar{S}Td^2)$ where $\bar{S}$ is the average region size, and requires a large amount of memory to store all the previous observations.

Instead, we derive a mean covariance matrix computation which does not require keeping all the previous measurements and solves the described problems. We construct and update a temporal kernel of covariance matrices. Let $\{\mathbf{C}_t\}_{t=1\ldots T}$ be the set of T previous covariance matrices corresponding to the previously estimated object regions $\{R_t\}_{t=1\ldots T}$. From this set, we compute a sample mean covariance matrix that blends all the previous matrices.

As discussed in Section 3.2.2, the Euclidean distance is not an appropriate metric for covariance matrices. For example, it is shown in [122] that the Euclidean mean of covariance matrices have larger determinant than the original matrix determinants.

The $d \times d$ dimensional symmetric positive definite matrices (nonsingular covariance matrices), Sym_{+}^{d}, can be formulated as a connected Riemannian manifold and an equivalent form of distance metric (3.3) is given in (A.34). See Section A.7.2 for more details. The mean of the points on Sym_{d}^{+} is the point on the manifold which minimizes the sum of squared Riemannian distances (A.34), $\sum_{t=1}^{T} d^2(\mathbf{C}, \mathbf{C}_t)$. The mean can be found by minimizing the error via a gradient descent algorithm [122]

$$\bar{\mathbf{C}}^{i+1} = \exp_{\bar{\mathbf{C}}^i} \left[\frac{1}{N} \sum_{t=1}^{T} \log_{\bar{\mathbf{C}}^i}(\mathbf{C}_t) \right] \tag{3.18}$$

	miss/total	detection	trials
Pool Player [1]	8/92	91.4	0.0356
Running Dog [1]	9/125	92.8	0.0284
Subway [1]	4/173	97.6	0.0091
Jogging [1]	20/824	97.7	0.0096
Street-color [1]	16/180	91.1	0.0351
Street-infrared [1]	61/180	66.2	0.1337
Street-joint [1]	8/180	95.6	0.0175
Race [2]	2/692	99.7	0.0015
Crowd [3]	7/522	99.1	0.0034

Table 3.3: Tracking performance scores. *Detection* is the percentage of correct estimation rate. *Trials* is the percentage of the number of trials to get a correct estimate to the total number of locations. Video sizes are 352×288^1, 352×240^2, 440×360^3

.

where exp and log operators are defined in (A.30) and (A.31) respectively. The method iterates by computing first order approximations to the mean on the tangent space.

In the above formulation, all the previous matrices $\{\mathbf{C}_t\}_{t=1...T}$ in the set are considered as equally influential on the result regardless of whether they are accurate or not. To prevent the model from contamination, it is possible to weight the data points proportional to its similarity to the current model. Then, the algorithm is simply modified as

$$\bar{\mathbf{C}}^{i+1} = \exp_{\bar{\mathbf{C}}^i}\left(\frac{1}{\rho^*}\sum_{t=1}^{T}\rho^{-1}(\mathbf{C}_t, \bar{\mathbf{C}}^*)\log_{\bar{\mathbf{C}}^i}(\mathbf{C}_t)\right). \tag{3.19}$$

where ρ is defined in (3.3), $\rho^* = \sum_{t=1}^{T}\rho^{-1}(\mathbf{C}_t, \bar{\mathbf{C}}^*)$ and $\bar{\mathbf{C}}^*$ is the model computed at the previous frame. Please see Section 4.3.2 for more details on sample means on Riemannian manifolds.

3.5.2 Tracking Experiments

We assessed the performance using 15 sequences totaling more than 3000 frames. These include moving and stationary camera recordings, infrared sequences, etc., and some of the results are listed in Table 3.3. We computed two performance metrics. The *detection rate* is the ratio of the number of frames the object location is accurately estimated to the total number of frames in the sequence. We consider the estimated location accurate if the best match is within the 9×9 neighborhood of the ground truth

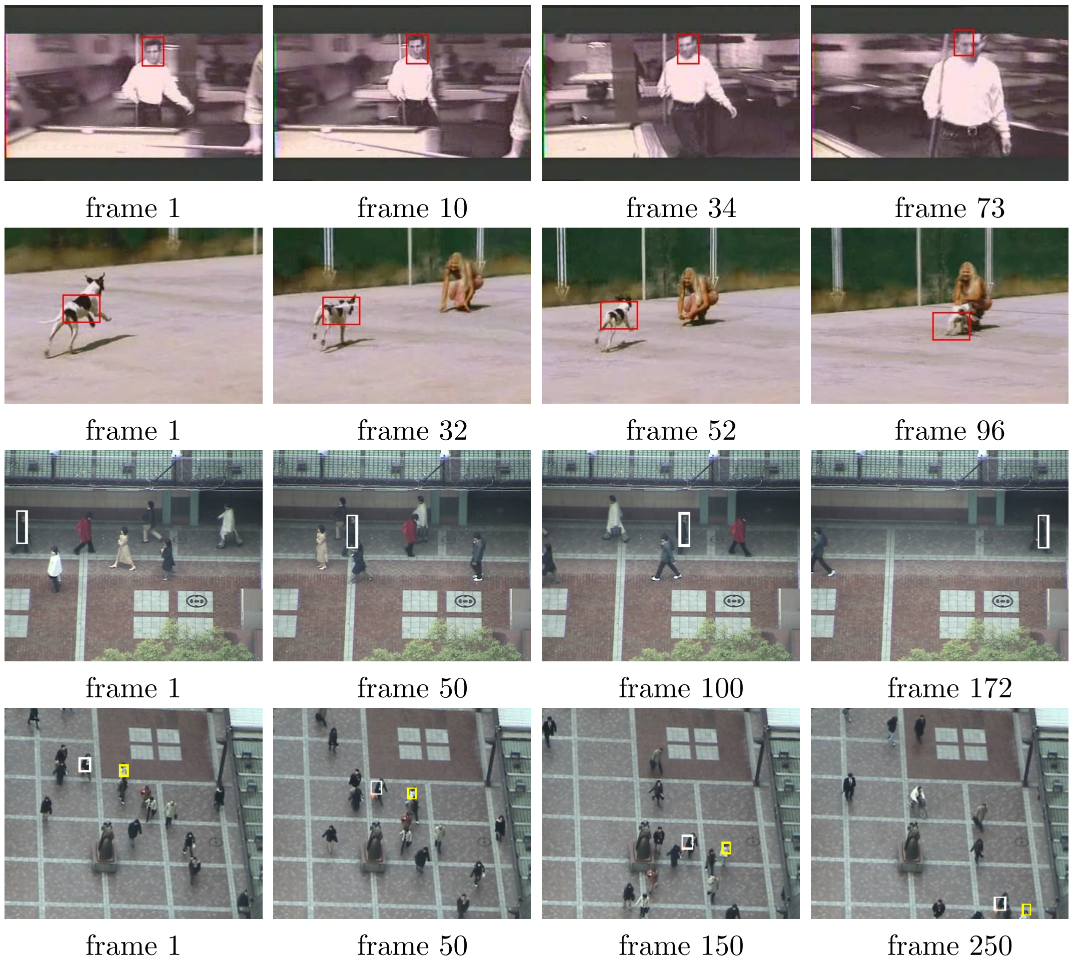


Figure 3.7: Tracking results for four different sequences. In *Pool Player* and *Running Dog* sequences, the camera and objects are moving, and the appearances are changing. In *Subway* and *Crowd*, the objects have indistinctive color and insignificant texture information.

object center location. For example, there are 101376 possible regions for a 352×288 image and the probability of correctly estimating the object location is $1 : 1251$, if we draw it randomly.

We also analyzed the *number of trials* to find the correct estimation. This is based on ordering the search regions according to the match scores until we find the correct estimation. We defined the metric as the percentage of the number of trials to the total number of locations.

Sample tracking results are given in Figures 3.7 and 3.8. For color sequences, we used all 3 RGB channels as separate features. For sequences recorded in stationary camera

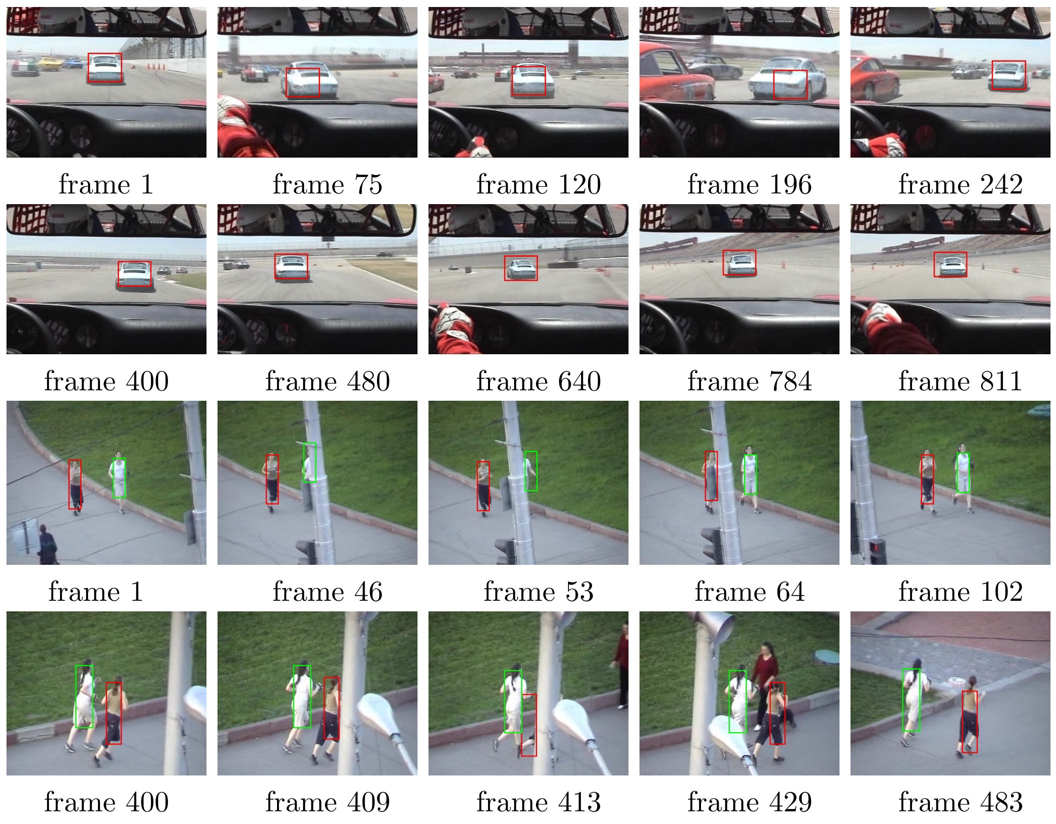


Figure 3.8: Tracking results using for moving camera sequences. Size changes (frames 75, 196, 242, 881) in *Race* sequence and severe occlusions (frames 53, 64, 409, 413) in *Jogging* sequence are accurately detected.

setups, we included a frame difference score. The frame difference feature improved the performance in infrared sequences since infrared imagery lack of sufficient spatial information to compute reliable features for small objects.

Objects are manually initialized and we applied the covariance tracking with the weighted mean based update mechanism. We computed the covariance matrices in full resolution feature image and performed the exhaustive search in half resolution grid to find the best match.

We observed that the covariance modeling and update mechanism successfully detect and adapt models to the undergoing changes as several examples are given in Figures 3.7 and 3.8. Note that, in approximately 1% of the frames the objects were fully occluded, therefore the overall detection rate was bounded at 99%. Still, the covariance tracker was able to find objects at 97.4% of the frames as given in the first column of Table 3.4. In comparison, optimal histogram matching could detect only 72.8% of objects in our

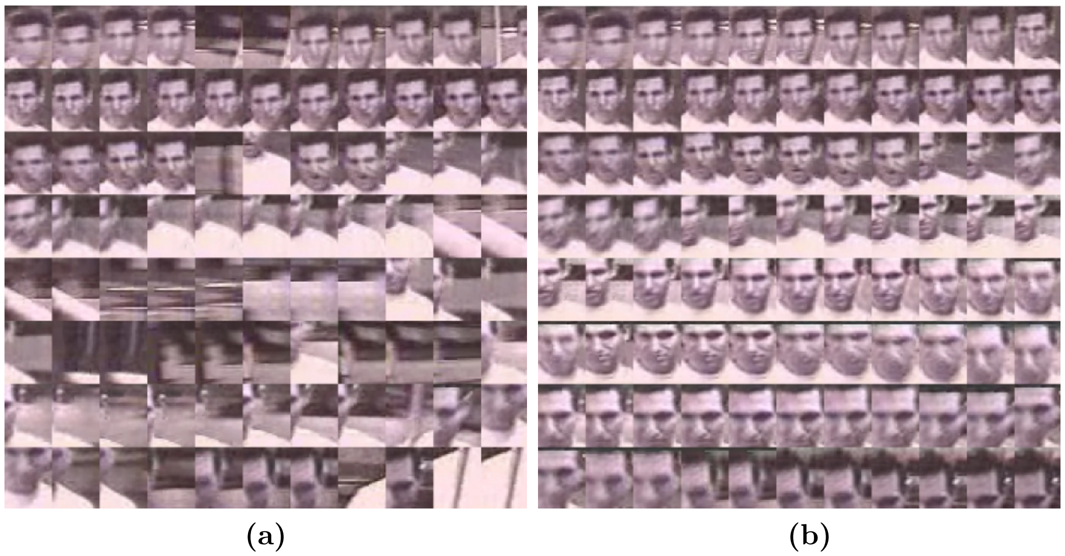


Figure 3.9: Montages of the detected results from 88 consecutive frames of *Pool Player* sequence. Some frames can be seen in Figure 3.7. (a) With no model update; detection rate is 47.7%. (b) With weighted mean based update mechanism; detection rate is 100%.

datasets. The original mean shift [29], on the other hand, was able to keep track of objects only for a couple of initial frames in case the objects move fast and erratically (*Jogging*) or the color variation is low and object color resembles to the background (*Pool Player*). The average tracking performance of the original mean shift was less than 40%.

Figure 3.9 shows sample results with and without model update. We observed that the model update becomes more critical especially for the objects having non-rigid deformations and pose changes. The model adaptation speed relies on the number of previous frames T. For example, $T = 5$ provides flexible models while $T = 40$ gives more robust estimates.

Fusion of Infrared and Color

The covariance matrix provides an effective solution to combine different modalities. By extending the feature vector to include the temperature values for pixel-wise aligned infrared and color sequences, we were able to take the advantage of both modalities. In Figure 3.10, detected objects are given for three cases; tracking using color only, infrared only, and joint as described. Detection rate has significantly improved from

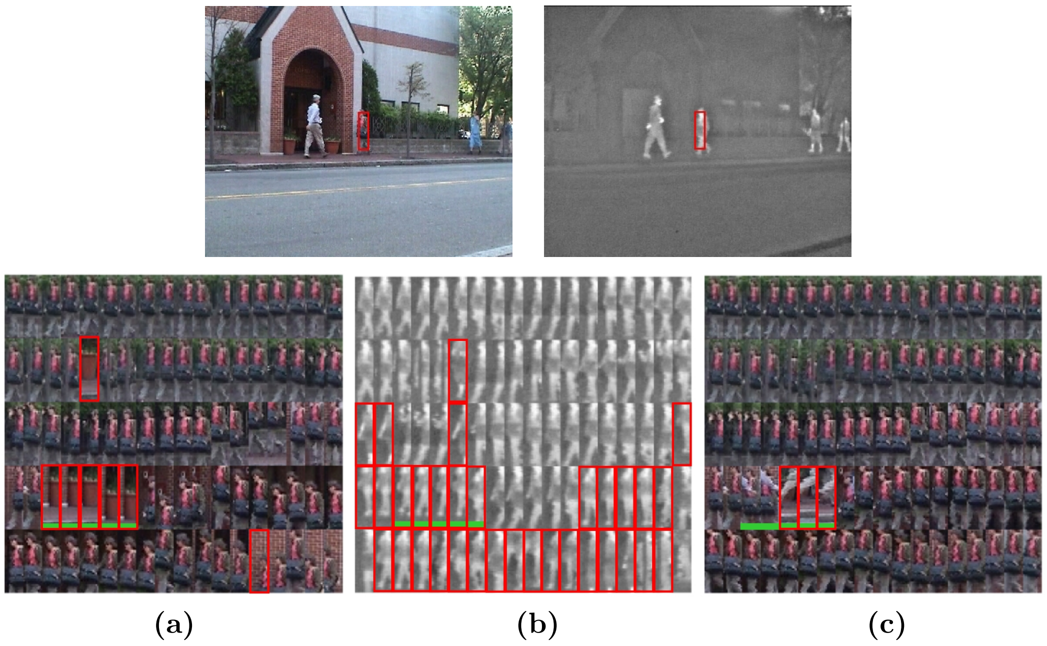


Figure 3.10: Detection rates of (a) color: 92%, (b) infrared: 60%, (c) joint: 96%. Note that, localization also improves. Red boxes in the montage images indicate the misses. Green indicates the frames where the object is fully occluded.

92%-color and 60%-infrared to 96%-joint, and the best matches became closer to the ground truth.

Noise and Illumination Changes

To test sensitivity against noise, we contaminated the color values with additive zero mean Gaussian noise with variance σ_η^2, where sample results are shown in Figure 3.11. We observed that while the performance of the histogram matching performance significantly degrades (down to 18.9%), the covariance tracking consistently achieves higher detection rates (94.3% to 70.6%), as given in Table 3.4. Although a common feature for tracking, histograms are easily contaminated by the noise and loose their saliency.

To analyze robustness against the illumination changes, we randomly scaled the color values of each frame as $I(x, y) = r_t I(x, y)$ where r_t is a random number between 0.2 and 1.0. The random numbers r_t, r_{t+1} were independent, thus sudden severe variations were allowed. The detection rates are given in Table 3.5. To be more robust toward illumination changes for histogram matching, we also tested hue-saturation values only. Still, the covariance tracking outperformed histogram matching. The last

σ_η^2	0	0.01	0.1	0.3
Covariance tr.	97.4	94.3	89.0	70.6
Histogram mat.	72.8	65.2	42.6	18.9

Table 3.4: Detection Rates - Gaussian Noise Contamination. Infrared sequences are not included.

	RGB	HS-only
Covariance tracking	95.6	93.3
Histogram matching	48.7	64.0

Table 3.5: Detection Rates - Severe Illumination Change.

row of Figure 3.11 shows sample illumination transformed images and the montage images of the tracked object. The covariance tracker is very robust against the sudden illumination changes.

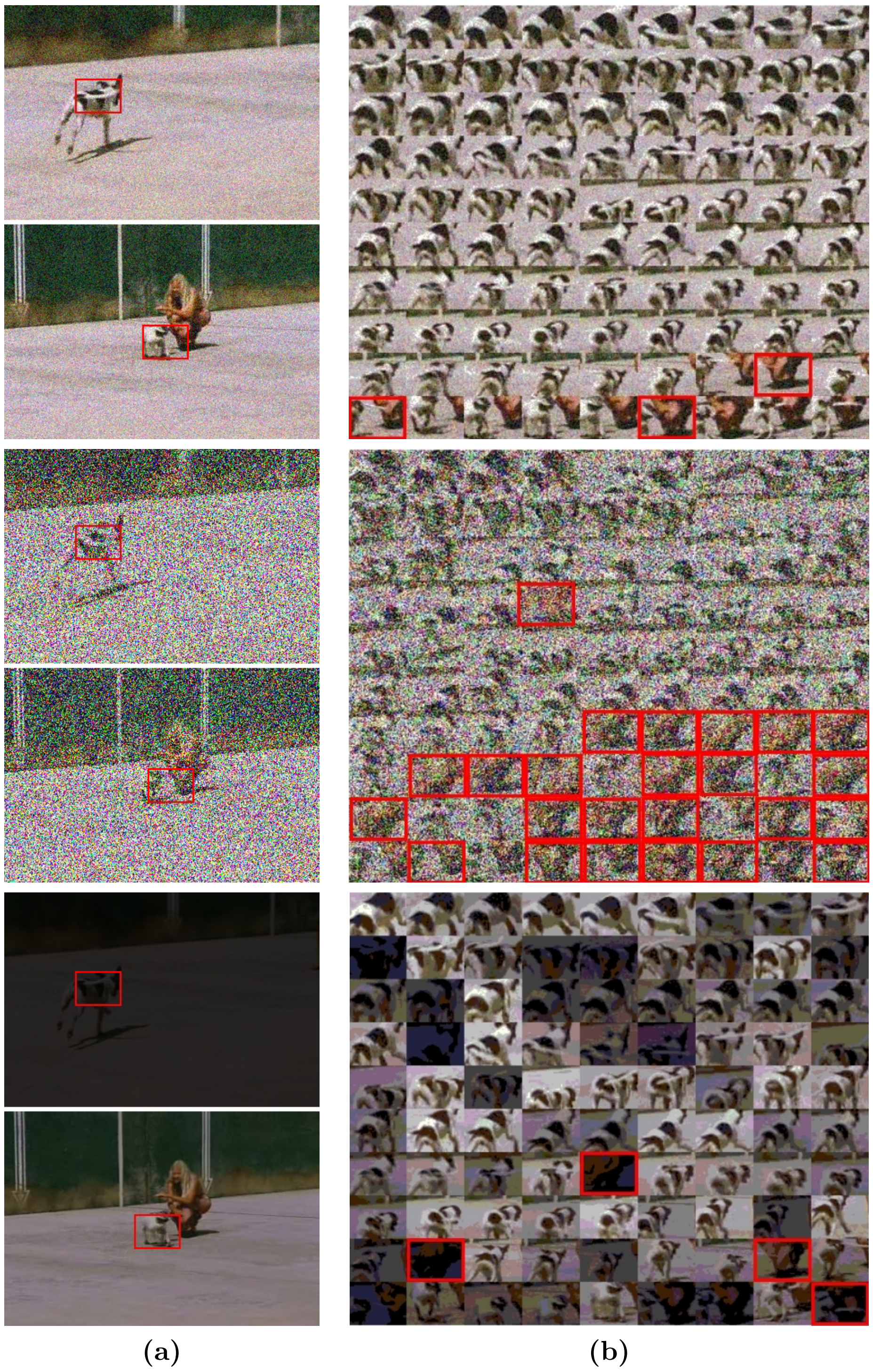


Figure 3.11: Tracking under noise and illumination change. (a) Frames 8 and 84 from *Running Dog*. (b) Montages of 90 detected locations. From top to bottom: noisy data with $\sigma_\eta^2 = 0.01$ (detection rate for this sequence is 96.6%), noisy data with $\sigma_\eta^2 = 0.3$ (detection rate of 68.9%), sudden light changes (detection rate of 95.6%). Red boxes in the montage images indicate the misses.

Chapter 4

Classification on Riemannian Manifolds

4.1 Introduction

Supervised learning or commonly referred as statistical classification is the problem of learning a method which can associate an unseen observation to an existing class based on a set of previously labeled items. With the advances in learning techniques and availability of labeled datasets, learning based methods become increasingly popular in the field of computer vision. Several important computer vision problems can be formulated as a supervised learning problem. Object detection, recognition, retrieval, tracking and image segmentation are only a few of the relevant examples.

A major shortcoming of most of the existing supervised learning techniques is that the methods assume that the domain of the classifiers form a vector space. As already discussed before, when the assumption is violated the methods either become inapplicable or significant performance degradation is observed. Here we present a novel approach for classifying points lying on a connected Riemannian manifold using the geometry of the space [155, 157]. Without loss of generality we describe the approach on the space of symmetric positive definite matrices, however the method generalizes to any connected Riemannian manifold. Although there have been previous approaches for clustering data points lying on differentiable manifolds [8, 148, 158], to our knowledge, this study is one of the earliest works aiming on the classification problem. We present an application of the derived algorithm for the pedestrian detection problem in still images where the region covariance features are utilized as object descriptors.

Detecting different categories of objects in image and video content is one of the fundamental tasks in computer vision research. The success of many applications such as visual surveillance, image retrieval, robotics, autonomous vehicles and smart cameras are conditioned on the accuracy of the detection process.

Two main processing steps can be distinguished in a typical object detection algorithm. The first task is the *feature extraction*, in which the most informative object descriptors regarding the detection process are obtained from the visual content. The second task is the *detection*, in which the obtained object descriptors are utilized in a classification framework to detect the objects of interest.

The feature extraction methods can be further categorized into two groups based on the representation. The first group of methods is the *sparse representations*, where a set of representative local regions are obtained as the result of an interest point detection algorithm. Reliable interest points should encapsulate valuable information about the local image content, and remain stable under changes such as viewpoint and/or illumination. There exists an extensive literature on interest point detectors, and [51, 68, 87, 100, 108] are only a few of the most commonly used methods which satisfy consistency over a large range of operating conditions.

Earlier approaches for part descriptors utilized intensity based features. However, histogram based representation of image gradients and edges in spatial context, such as scale invariant feature transform (SIFT) descriptors [100] or shape contexts [9], were shown to yield more robust and distinctive descriptors. Several object detection algorithms were proposed by assembling the detected parts according to spatial relationships in probabilistic frameworks [47, 169], by discriminative approaches [1, 119] or via matching shapes [10, 99].

The second group of feature extraction methods is the *dense representations*, where object descriptors are obtained inside a detection window. The entire image is scanned densely (possibly each pixel), and a learned classifier of the object model is evaluated. Earlier studies utilized image intensities such as intensity templates [134, 149], or principal component analysis (PCA) coefficients [145, 152] to represent the object model. More recently, Haar wavelet based descriptors which are a set of basis functions encoding intensity differences between two regions, became increasingly popular due to efficient computation and superiority to encode visual patterns. In [121], an overcomplete dictionary of basis functions were computed from overlapping regions utilizing horizontal, vertical and diagonal features inside the detection window. Instead of sampling among an overcompete dictionary of features, in [165], a small set of important features were selected via a greedy selection method using AdaBoost.

Most of the leading approaches in object detection are discriminative methods such as neural networks (NNs) [72], support vector machines (SVMs) [31] or boosting [137]. These methods became increasingly popular since they can cope with high dimensional state spaces and/or are able to select relevant descriptors among a large set. In [134, 149] NNs, and in [121] SVMs were utilized as a single strong classifier for detection of various categories of objects. The NNs and SVMs were also utilized for intermediate representations [38, 115] for final object detectors. In [165], multiple weak classifiers trained using AdaBoost were combined to form a rejection cascade, such that if any classifier rejects a hypothesis, then it is considered a negative example. The approach provides an efficient algorithm due to sparse feature selection, besides only a few classifiers are evaluated at most of the regions due to the cascade structure. Variants of the algorithm were also proposed to share features among different object categories, thereby providing a more efficient solution for multiple class object detection [150]. The other approaches include the probabilistic methods [47, 169], where the conditional densities of object and non-object classes are modeled explicitly.

Pedestrian detection in still images is considered among the hardest examples of object detection problems. The articulated structure and variable appearance of the human body, combined with illumination and pose variations, contribute to the complexity of the problem.

Sparse representations for pedestrian detection include models for detecting body parts [79, 131] or common shapes [107], where these local features were assembled according to geometric constraints to form the final pedestrian model. In [109], parts were represented by co-occurrences of local orientation features, and separate detectors were trained for each part using AdaBoost. Target location was determined by maximizing the joint likelihood of part occurrences combined according to the geometric relations. A pedestrian detection system for crowded scenes was described in [96]. The approach combined local appearance features and their geometric relations with global cues by top-down segmentation based on per pixel likelihoods. Other approaches include using silhouette information either in matching [59] or in classification framework [120].

Dense representations for pedestrian detection include [46], where a cost function is defined based on the part likelihoods and their joint configuration. The minimizer of the function with respect to all the possible part locations in the image plane is efficiently found using dynamic programming. In [121], a polynomial SVM was learned using Haar wavelets as pedestrian descriptors. Later, the work was extended to multiple classifiers trained to detect human parts, and the responses inside the detection window were combined to give the final decision [113]. Similar to still images, in [166], a real time moving pedestrian detection algorithm was described also using Haar wavelet descriptors, but extracted from space-time differences in video. Using AdaBoost, the most discriminative features were selected, and multiple classifiers were combined to form a rejection cascade. In [36], an excellent pedestrian detector was described by training an SVM classifier using densely sampled histogram of oriented gradients (similar to SIFT descriptors) inside the detection window. The performance of the proposed descriptors was shown on INRIA human dataset. In a similar approach [181], near real time detection performances were achieved by training a cascade model using histogram of oriented gradients (HOG) features. Recently in [136], a two stage AdaBoost classifier is learned using shapelet features and comparable performances are reported on INRIA human dataset. The initial stage learns a set of classifiers which are a combination of oriented gradient responses, whereas the second stage combines the classifier responses to form the final detector. We refer to [35] for a detailed survey on object and pedestrian detection methods.

Here we present a dense model where covariance features are utilized as pedestrian descriptors inside a detection window. We represent a pedestrian with several covariance descriptors of overlapping regions where the best descriptors are determined with a greedy feature selection algorithm combined with boosting. The classifiers are then learned on the tangent spaces of Sym_d^+.

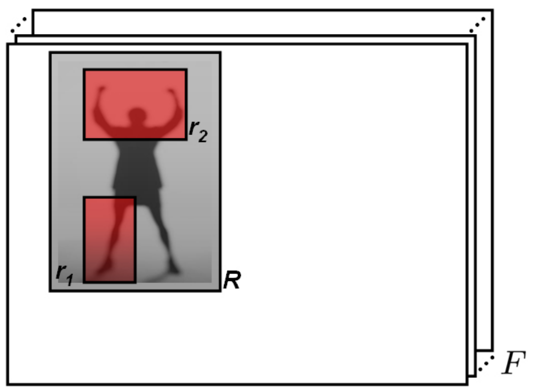


Figure 4.1: The detection window is R and r_1, r_2 are two possible descriptor subwindows.

4.2 Covariance Descriptors for Pedestrian Detection

The covariance descriptors were described in Chapter 3. Here we describe the customization of the descriptors for pedestrian detection problem.

We define the feature mapping (3.1), $\Phi(I, x, y)$, as

$$F(x,y) = \left[x \quad y \quad |I_x| \quad |I_y| \quad \sqrt{I_x^2 + I_y^2} \quad |I_{xx}| \quad |I_{yy}| \quad \arctan \frac{|I_x|}{|I_y|} \right]^T \tag{4.1}$$

where x and y are pixel location, $I_x, I_{xx}, ..$ are intensity derivatives and the last term is the edge orientation. With the defined mapping, the input image is mapped to a $d = 8$ dimensional feature image. The covariance descriptor of a region is an 8×8 matrix, and due to symmetry only upper triangular part is stored, which has only 36 different values. The descriptor encodes information of the variances of the defined features inside the region, their correlations with each other and spatial layout.

Given an arbitrary sized detection window R, there are a very large number of covariance descriptors that can be computed from subwindows $r_{1,2,\ldots}$, as shown in Figure 4.1. We perform sampling and consider all the subwindows r starting with minimum size of 1/10 of the width and height of the detection window R, at all possible locations. The size of r is incremented in steps of 1/10 along the horizontal or vertical, or both, until $r = R$. Although the approach might be considered redundant due to overlaps, there is significant evidence that the overlapping regions are an important factor in detection performances [36, 181]. The greedy feature selection mechanism, that will be described later, allows us to search for the best regions during learning classifiers.

Although it has been mentioned that the covariance descriptors are robust towards illumination changes, we would like to enhance the robustness to also include local illumination variations in an image. Let r be a possible feature subwindow inside the detection window R. We compute the covariance of the detection window $\mathbf{C}_R$ and subwindow $\mathbf{C}_r$ using integral representation. The normalized covariance descriptor of region r, $\hat{\mathbf{C}}_r$, is computed by dividing the columns and rows of $\mathbf{C}_r$ with the square root of the respective diagonal entries of $\mathbf{C}_R$,

$$\hat{\mathbf{C}}_r = diag(\mathbf{C}_R)^{-\frac{1}{2}}\mathbf{C}_r diag(\mathbf{C}_R)^{-\frac{1}{2}} \tag{4.2}$$

where $diag(\mathbf{C}_R)$ is equal to $\mathbf{C}_R$ at the diagonal entries and the rest is truncated to zero. The method described is equivalent to first normalizing the feature vectors inside the region R to have zero mean and unit standard deviation, and after that computing the covariance descriptor of subwindow r. Notice that under the transformation, $\hat{\mathbf{C}}_R$ is equal to the correlation matrix of the features inside the region R. The process only requires d^2 extra division operations.

4.3 Riemannian Geometry

A brief introduction to Riemannian geometry is given in Appendix A. The $d \times d$ dimensional symmetric positive definite matrices (nonsingular covariance matrices), Sym_+^d, can be formulated as a connected Riemannian manifold. Please see Section A.7.2 for details. Here we define the necessary operators on the manifold for the classification algorithm.

4.3.1 Orthonormal Coordinates for the Tangent Space

For classification, we need a minimal representation of the points on the tangent space. In general, the standard coordinate system induced by the chart maps is not orthonormal with respect to the Riemannian metric at all the points on the manifold. Even though the basis can be orthonormal for some points on the manifold, it should be adjusted for the Riemannian metric for the other locations. The orthonormal coordinate system for the tangent space is defined by the vector operator, $\text{vec}_{\mathbf{X}}$, which gives the orthonormal coordinates of a tangent vector at $T_{\mathbf{X}}$.

For Sym_d^+, the tangent space is the space of symmetric matrices and there are only $m = d(d+1)/2$ independent coefficients which are the upper triangular or the lower triangular part of the matrix. First we define the orthonormal basis for the tangent space at identity. Let $\mathbf{y}$ be a vector on the tangent space at identity. The norm of the vector is given by $< \mathbf{y}, \mathbf{y} >_{\mathbf{I}}$, where the inner product is defined in (A.29). The off-diagonal entries are counted twice during the norm computation. Therefore, the orthonormal coordinates of a tangent $\mathbf{y} \in T_{\mathbf{I}}$ is given by

$$\text{vec}_{\mathbf{I}}(\mathbf{y}) = [y_{1,1} \ \sqrt{2}y_{1,2} \ \sqrt{2}y_{1,3} \ \ldots \ y_{2,2} \ \sqrt{2}y_{2,3} \ \ldots \ y_{d,d}]^T. \tag{4.3}$$

The vector operator, $\text{vec}_{\mathbf{I}}$, relates the Riemannian metric (A.29) at $T_{\mathbf{I}}$ to the canonical metric defined in $\mathbb{R}^m$

$$< \mathbf{y}, \mathbf{y} >_{\mathbf{I}} = \|\text{vec}_{\mathbf{I}}(\mathbf{y})\|_2^2. \tag{4.4}$$

Let $\mathbf{y} \in T_{\mathbf{X}}$. The orthonormal coordinates of a tangent vector $\mathbf{y}$ at $T_{\mathbf{X}}$ is then given by

$$\text{vec}_{\mathbf{X}}(\mathbf{y}) = \text{vec}_{\mathbf{I}}(\mathbf{X}^{-\frac{1}{2}}\mathbf{y}\mathbf{X}^{-\frac{1}{2}}). \tag{4.5}$$

It can be easily verified that the vector operator, $\text{vec}_{\mathbf{X}}$, satisfies

$$< \mathbf{y}, \mathbf{y} >_{\mathbf{X}} = \|\text{vec}_{\mathbf{X}}(\mathbf{y})\|_2^2. \tag{4.6}$$

Notice that, the tangent vector $\mathbf{y}$ is a symmetric matrix, and with the vector operator, $\text{vec}_{\mathbf{X}}(\mathbf{y})$, we get the orthonormal coordinates of $\mathbf{y}$ which is in $\mathbb{R}^m$.

4.3.2 Mean of the Points on Riemannian Manifolds

Let $\{\mathbf{X}_i\}_{i=1\ldots N}$ be the set of points on a Riemannian manifold $\mathcal{M}$. Similar to Euclidean spaces, the Karcher mean [90] of points on Riemannian manifold, is the point on $\mathcal{M}$ which minimizes the sum of squared Riemannian distances (A.11),

$$\boldsymbol{\mu} = \arg \min_{\mathbf{X} \in \mathcal{M}} \sum_{i=1}^{N} d^2(\mathbf{X}_i, \mathbf{X}) \tag{4.7}$$

where in our case, Sym_d^+, d^2 is the distance metric (A.34). In general, the Riemannian mean for a set of points is not necessarily unique. This can be easily verified by considering two points at antipodal positions on a sphere, where the error function is minimal for any point lying on the equator. However, it is shown in several studies that the mean is unique and the gradient descent algorithm is convergent for Sym_d^+ [49][112][122].

Differentiating the error function with respect to $\mathbf{X}$, we see that mean is the solution to the nonlinear matrix equation

$$\sum_{i=1}^{N} \log_{\mathbf{X}}(\mathbf{X}_i) = 0 \tag{4.8}$$

which can be solved iteratively with the following gradient descent procedure [122]

$$\boldsymbol{\mu}^{t+1} = \exp_{\boldsymbol{\mu}^t}\left[\frac{1}{N}\sum_{i=1}^{N} \log_{\boldsymbol{\mu}^t}(\mathbf{X}_i)\right]. \tag{4.9}$$

The method iterates by computing first order approximations to the mean on the tangent space. The weighted mean computation is similar to (4.9). We replace inside of the exponential, the mean of the tangent vectors, with the weighted mean $\frac{1}{\sum_{i=1}^{N} w_i}\sum_{i=1}^{N} w_i \log_{\boldsymbol{\mu}^t}(\mathbf{X}_i)$.

4.4 Classification on Riemannian Manifolds

Let $\{(\mathbf{X}_i, y_i)\}_{i=1\ldots N}$ be the training set with respect to class labels, where $\mathbf{X}_i \in \mathcal{M}$, $y_i \in \{0, 1\}$ and $\mathcal{M}$ is a Riemannian manifold. We want to find a function $F(\mathbf{X}) : \mathcal{M} \mapsto \{0, 1\}$, which divides the manifold into two based on the training set of labeled items.

A function which divides the manifold is rather a complicated notion compared with the Euclidean space. For example, consider the simplest form a linear classifier in $\mathbb{R}^2$. A point and a direction vector in $\mathbb{R}^2$ define a line which separates $\mathbb{R}^2$ into two. Equivalently, on a two-dimensional differentiable manifold, we can consider a point on the manifold and a tangent vector in the tangent space of the point, which together define a curve on the manifold via exponential map. For example, if we consider the image of the lines on the 2-torus, the curves never divide the manifold into two.

A straightforward approach for classification would be to map the manifold to a higher dimensional Euclidean space, which can be considered as flattening the manifold.

However in a general case, there is no such mapping that globally preserves the distances between the points on the manifold. Therefore a classifier trained on the flattened space does not reflect the global structure of the points.

4.4.1 Local Maps and Boosting

We propose an incremental approach by training several weak classifiers on the tangent spaces, and combining them through boosting. We start by defining mappings from neighborhoods on the manifold to the Euclidean space, similar to coordinate charts. Our maps are the logarithm maps, $\log_{\mathbf{X}}$, that map the neighborhood of points $\mathbf{X} \in \mathcal{M}$ to the tangent spaces $T_{\mathbf{X}}$. Since this mapping is a homeomorphism around the neighborhood of the point, the structure of the manifold is locally preserved. The tangent space is a vector space, and we use standard machine learning techniques to learn the classifiers on this space.

For classification task, the approximations to the Riemannian distances computed on the ambient space should be as close to the true distances as possible. Let $\mathbf{Y}, \mathbf{Z} \in \mathcal{M}$. Since we approximate the distances (A.11) on the tangent space, $T_{\mathbf{X}}$,

$$d^2(\mathbf{Y}, \mathbf{Z}) \approx \|\mathrm{vec}_{\mathbf{X}}(\log_{\mathbf{X}}(\mathbf{Z})) - \mathrm{vec}_{\mathbf{X}}(\log_{\mathbf{X}}(\mathbf{Y}))\|_2^2 \tag{4.10}$$

is a first order approximation. The approximation error can be expressed in terms of the pairwise distances computed on the manifold and the tangent space

$$\epsilon^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \left(d(\mathbf{X}_i, \mathbf{X}_j) - \|\mathrm{vec}_{\mathbf{X}}(\log_{\mathbf{X}}(\mathbf{X}_i)) - \mathrm{vec}_{\mathbf{X}}(\log_{\mathbf{X}}(\mathbf{X}_j))\|_2\right)^2 \tag{4.11}$$

which is equal to

$$\begin{aligned}\epsilon^2_{Sym_d^+} = \sum_{i=1}^{N}\sum_{j=1}^{N} \quad & \left(\left\|\log\left(\mathbf{X}_i^{-\frac{1}{2}}\mathbf{X}_j\mathbf{X}_i^{-\frac{1}{2}}\right)\right\|_F - \right. \\ & \left. \left\|\log\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{X}_i\mathbf{X}^{-\frac{1}{2}}\right) - \log\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{X}_j\mathbf{X}^{-\frac{1}{2}}\right)\right\|_F \right)^2\end{aligned} \tag{4.12}$$

for the space of symmetric positive definite matrices using (A.31) and (4.6).

The classifiers can be learned on the tangent space at any point $\mathbf{X}$ on the manifold. Best approximation, which preserves the pairwise distances is achieved at the minimum of $\epsilon^2_{Sym_d^+}$. The error can be minimized with respect to $\mathbf{X}$ which gives the best tangent space to learn the classifier.

Since the mean of the points (4.7) is the minimizer of the sum of squared distances from the points in the set and the mapping preserves the structure of the manifold locally, it is also a good candidate for the minimizer of the error function (4.12). However, to our knowledge a theoretical proof does not exist. For some special cases it can be easily verified that the mean is the minimizer. Such a case arises when all the points lie on a geodesic curve, where the approximation error is zero for any point lying on the curve. Since mean also lies on the geodesic curve, the approximation is perfect. Nevertheless, for a general set of points, we only have empirical validation based on simulations. We generated random points on Sym_d^+. The approximation errors were measured on the tangent spaces at any of the points $\{T_{\mathbf{X}_i}\}_{i=1...N}$ and at the mean T_μ. In our simulations, the errors computed on the tangent spaces at the means were significantly lower than any other choice and counter examples were not observed. The simulations were also repeated for weighted sets of points, where the minimizers of the weighted approximation errors were achieved at the weighted means of the points.

Therefore, the weak learners are learned on the tangent space at the mean of the points. At each iteration, we compute the weighted mean of the points through (4.9), where the weights are adjusted through boosting. Then we map the points to the tangent space at the weighted mean and learn a weak classifier on this vector space. Since the weights of the samples which are misclassified during the earlier stages of boosting increase, the weighted mean moves towards these points and more accurate classifiers are learned for these points. The process is illustrated in Figure 4.2. To evaluate a test example, the sample is projected to the tangent spaces at the computed weighted means, and the weak learners are evaluated (Figure 4.3). The approximation error is minimized by averaging over several weak learners.

4.4.2 LogitBoost on Riemannian Manifolds

The LogitBoost algorithm [54] is the instance of the boosting algorithm that we utilize for our detector. We start with a brief description of LogitBoost algorithm on vector

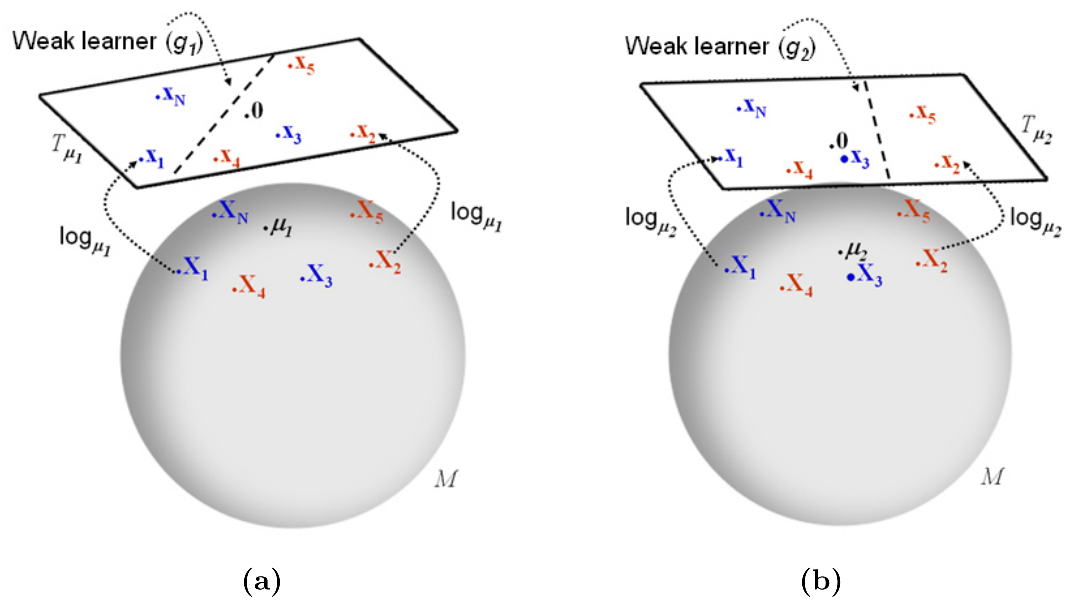


Figure 4.2: Two iterations of boosting on a Riemannian manifold. The manifold is depicted with the surface of the sphere and the plane is the tangent space at the mean. The samples are projected to tangent spaces at means via $\log_{\boldsymbol{\mu}_l}$. The weak learners g_l are learned on the tangent spaces $T_{\boldsymbol{\mu}_l}$. In (a), sample $\mathbf{X}_3$ is misclassified therefore its weight increases. In the second iteration of boosting (b), the weighted mean moves towards $\mathbf{X}_3$.

spaces. We consider the binary classification problem, $y_i \in \{0,1\}$. The probability of $\mathbf{x}$ being in class 1 is represented by

$$p(\mathbf{x}) = \frac{e^{F(\mathbf{x})}}{e^{F(\mathbf{x})} + e^{-F(\mathbf{x})}} \qquad F(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{L} f_l(\mathbf{x}). \tag{4.13}$$

The LogitBoost algorithm learns the set of regression functions $\{f_l(\mathbf{x})\}_{l=1\ldots L}$ (weak learners) by minimizing the negative binomial log-likelihood of the data $l(y, p(\mathbf{x}))$

$$-\sum_{i=1}^{N} [y_i log(p(\mathbf{x}_i)) + (1-y_i) log(1-p(\mathbf{x}_i))] \tag{4.14}$$

through Newton iterations. At the core of the algorithm, LogitBoost fits a weighted least square regression, $f_l(\mathbf{x})$ of training points $\mathbf{x}_i \in \mathbb{R}^m$ to response values $z_i \in \mathbb{R}$ with weights w_i where

$$z_i = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)(1-p(\mathbf{x}_i))} \qquad w_i = p(\mathbf{x}_i)(1-p(\mathbf{x}_i)). \tag{4.15}$$

The LogitBoost algorithm on Riemannian manifolds is similar to the original LogitBoost, except a few differences at the level of weak learners. In our case, the domain of

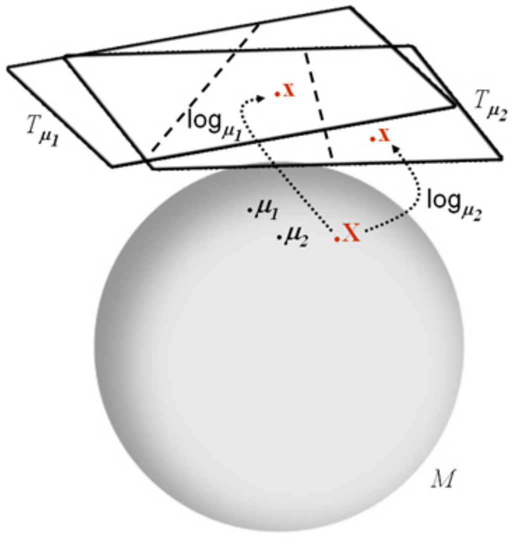


Figure 4.3: Classification on a Riemannian manifold. The sample $\mathbf{X}$ is projected to the tangent spaces $T_{\boldsymbol{\mu}_l}$ and the weak learners are evaluated.

the weak learners are in $\mathcal{M}$ such that $f_l(\mathbf{X}) : \mathcal{M} \mapsto \mathbb{R}$. Following the discussion of the previous section, we learn the regression functions on the tangent space at the weighted mean of the points. We define the weak learners as

$$f_l(\mathbf{X}) = g_l(\text{vec}_{\boldsymbol{\mu}_l}(\log_{\boldsymbol{\mu}_l}(\mathbf{X}))) \tag{4.16}$$

and learn the functions $g_l(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}$ and the weighted mean of the points $\boldsymbol{\mu}_l \in \mathcal{M}$. Notice that the mapping $\text{vec}_{\boldsymbol{\mu}_l}$ (4.5), gives the orthonormal coordinates of the tangent vectors in $T_{\boldsymbol{\mu}_l}$.

The algorithm is presented in Figure 4.4. The steps marked with (*) are the differences from original LogitBoost algorithm. For functions $\{g_l\}_{l=1\ldots L}$, it is possible to use any form of weighted least squares regression such as linear functions, regression stumps, etc., since the domain of the functions are in $\mathbb{R}^m$.

4.5 Pedestrian Detection

In this section, we describe the utilization of the LogitBoost algorithm on Riemannian manifolds for pedestrian detection problem. The domain of the classifier is the space of eight dimensional symmetric positive definite matrices, Sym_8^+. We combine $K = 30$

```
**Input:** Training set $\{(\mathbf{X}_i, y_i)\}_{i=1...N}$, $\mathbf{X}_i \in \mathcal{M}$, $y_i \in \{0,1\}$
  • Start with weights $w_i = 1/N$, $i = 1...N$, $F(\mathbf{X}) = 0$ and $p(\mathbf{X}_i) = \frac{1}{2}$
  • Repeat for $l = 1...L$
      – Compute the response values and weights
        $z_i = \frac{y_i - p(\mathbf{X}_i)}{p(\mathbf{X}_i)(1-p(\mathbf{X}_i))}$, $w_i = p(\mathbf{X}_i)(1 - p(\mathbf{X}_i))$
      – Compute weighted mean of the points through (4.9)
        $\boldsymbol{\mu}_l = \arg\min_{\mathbf{X}\in\mathcal{M}} \sum_{i=1}^{N} w_i d^2(\mathbf{X}_i, \mathbf{X})$ (*)
      – Map the data points to the tangent space at $\boldsymbol{\mu}_l$
        $\mathbf{x}_i = \mathrm{vec}_{\boldsymbol{\mu}_l}(\log_{\boldsymbol{\mu}_l}(\mathbf{X}_i))$ (*)
      – Fit the function $g_l(\mathbf{x})$ by weighted least-square regression of $z_i$ to $\mathbf{x}_i$ using
        weights $w_i$
      – Update $F(\mathbf{X}) \leftarrow F(\mathbf{X}) + \frac{1}{2} f_l(\mathbf{X})$ where $f_l$ is defined in (4.16) and
        $p(\mathbf{X}) \leftarrow \frac{e^{F(\mathbf{X})}}{e^{F(\mathbf{X})} + e^{-F(\mathbf{X})}}$
  • Store $F = \{\boldsymbol{\mu}_l, g_l)\}_{l=1...L}$
  • Output the classifier
    $\mathrm{sign}[F(\mathbf{X})] = \mathrm{sign}[\sum_{l=1}^{L} f_l(\mathbf{X})]$
```

Figure 4.4: LogitBoost on Riemannian manifolds.

LogitBoost classifiers on Sym_8^+ with rejection cascade, as shown in Figure 4.5. Weak learners $g_{k,l}$ are linear regression functions learned on the tangent space of Sym_8^+. The tangent space is $m = 36$ dimensional vector space.

To avoid confusion with Section 4.4.2, we refer to the training set by $\{(R_i, y_i)\}_{i=1...N}$, where R_i are the image windows containing background and pedestrians, and $y_i \in \{0,1\}$ are the labels. A very large number of covariance descriptors can be computed from a single detection window R. Therefore, we do not have a single set of positive and negative features, but several sets corresponding to each of the possible subwindows. Each weak learner is associated with a single subwindow of the detection window. Let $r_{k,l}$ be the subwindow associated with l-th weak learner of cascade level k. The normalized covariance descriptor of the i-th training sample associated with region $r_{k,l}$ is referred by $\mathbf{X}_{i,k,l} = \hat{\mathbf{C}}_{i,r_{k,l}}$. For simplicity, we use the shorthand notation

$$f_{k,l}(R_i) = f_{k,l}(\mathbf{X}_{i,k,l}) \tag{4.17}$$

for weak learners.

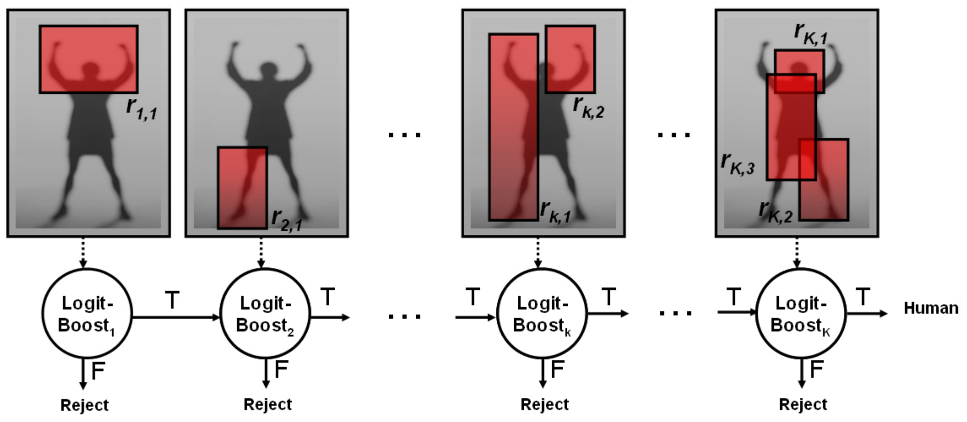


Figure 4.5: Cascade of LogitBoost classifiers. The kth LogitBoost classifier selects normalized covariance descriptors of subwindows $r_{k,i}$.

Let R_i^+ and R_i^- refer to the N_p positive and N_n negative samples in the training set, where $N = N_p + N_n$. While training the k-th cascade level, we classify all the negative examples $\{R_i^-\}_{i=1\ldots N_n}$ with the cascade of the previous $(k-1)$ LogitBoost classifiers. The samples which are correctly classified (samples classified as negative) are removed from the training set. Any window sampled from a negative image is a negative example, therefore the cardinality of the negative set, N_n, is very large. During training of each cascade level, we sample 10000 negative examples.

The learning algorithm is slightly customized for pedestrian detection task. We do not have a fixed number of weak learners L for each LogitBoost classifier k, but a variable number L_k. Each cascade level is optimized to correctly detect at least 99.8% of the positive examples, while rejecting at least 35% of the negative examples. In addition, we enforce a margin constraint between the positive samples and the decision boundary. Let $p_k(R)$ be the learned probability function of a sample being positive at cascade level k, evaluated through (4.13). Let R_p be the positive example that has the $(0.998N_p)$-th largest probability among all the positive examples. Let R_n be the negative example that has the $(0.35N_n)$-th smallest probability among all the negative examples. We continue to add weak classifiers to cascade level k until $p_k(R_p) - p_k(R_n) > margin$, where we set $margin = 0.2$. When the constraint is satisfied, the threshold (decision boundary) for cascade level k is stored as $thrd_k = F_k(R_n)$.

A test sample is classified as positive by cascade level k if $F_k(R) > thrd_k$ or equivalently $p_k(R) > p_k(R_n)$. With the proposed method, any of the positive training samples in the top 99.8 percentile have at least *margin* more probability than the points on the decision boundary. The process continues with the training of $(k+1)$-th cascade level, until $k = K$.

We incorporate a greedy feature selection method to produce a sparse set of classifiers focusing on important subwindows. At each boosting iteration l of the k-th LogitBoost level, we sample 200 subwindows among all the possible subwindows, and construct normalized covariance descriptors. We learn the weak classifiers representing each subwindow, and add the best classifier which minimizes the negative binomial log-likelihood (4.14) to the cascade level k. The procedure iterates with the training the $(l+1)$-th weak learner until the specified detection rates are satisfied.

The negative sample set is not well characterized for detection tasks. Therefore, while projecting the points to the tangent space, we compute the weighted mean of only the positive samples. Although rarely happens, if some of the features are fully correlated, there will be singularities in the covariance descriptor. We ignore those cases by adding very small identity matrix to the covariance. The pedestrian detection with cascade of LogitBoost classifiers on Sym_8^+ is given in Figure 4.6.

The learning algorithm produces a set of K LogitBoost classifiers which are composed of L_k triplets

$$F_k = \left\{(r_{k,l}, \boldsymbol{\mu}_{k,l}, g_{k,l})\right\}_{l=1\dots L_k} \quad \text{and} \quad thrd_k \tag{4.18}$$

where $r_{k,l}$ is the selected subwindow, $\boldsymbol{\mu}_{k,l}$ is the mean and $g_{k,l}$ is the learned regression function of the l-th weak learner of the k-th cascade. To evaluate a *test region* R with k-th classifier, the normalized covariance descriptors constructed from regions $r_{k,l}$ are projected to tangent spaces $T_{\boldsymbol{\mu}_{k,l}}$ and the features are evaluated with $g_{k,l}$

$$\text{sign}\left[F_k(R) - thrd_k\right] = \text{sign}\left[\sum_{l=1}^{L_k} g_{k,l}\left(\text{vec}_{\boldsymbol{\mu}_{k,l}}\left(\log_{\boldsymbol{\mu}_{k,l}}\left(\hat{\mathbf{C}}_{r_{k,l}}\right)\right)\right) - thrd_k\right]. \tag{4.19}$$

The initial levels of the cascade are learned on relatively easy examples, thus there are very few weak classifiers in these levels. Due to the cascade structure, only a few

```
Input: Training set $\{(R_i, y_i)\}_{i=1...N}$, $y_i \in \{0,1\}$
  • Repeat for $k = 1...K$
      – Classify negative examples $\{R_i^-\}_{i=1...N_n}$ with the cascade of $(k-1)$ classifiers and remove samples which are correctly (negative) classified
      – Initial weights $w_i = 1/N$, $i = 1...N$, $F_k(R) = 0$ and $p_k(R_i) = \frac{1}{2}$, let $l = 1$
      – Repeat while $p_k(R_p) - p_k(R_n) < margin$
          ∗ Compute the response values and weights
            $z_i = \frac{y_i - p_k(R_i)}{p_k(R_i)(1-p_k(R_i))}$, $w_i = p_k(R_i)(1 - p_k(R_i))$
          ∗ Sample $\{r_{k,t}\}_{t=1...200}$ subwindows and construct normalized covariance descriptors $\mathbf{X}_{i,k,t} = \hat{\mathbf{C}}_{i,r_{k,t}}$
          ∗ Repeat for $t = 1...200$
              · Compute the weighted mean of the positive samples $\{X_{i,k,t}^+\}_{i=1...N_p}$ through (4.9)
                $\boldsymbol{\mu}_{k,t} = \arg\min_{\mathbf{X} \in Sym_8^+} \sum_{i=1}^{N_p} w_i d^2(\mathbf{X}_{i,k,t}^+, \mathbf{X})$
              · Map the data points to the tangent space at $\boldsymbol{\mu}_{k,t}$
                $\mathbf{x}_{i,k,t} = \mathrm{vec}_{\boldsymbol{\mu}_t}(\log_{\boldsymbol{\mu}_t}(\mathbf{X}_{i,k,t}))$
              · Fit function $g_{k,t}(\mathbf{x})$ by weighted least-squares regression of $z_i$ to $\mathbf{x}_{i,k,t}$ using weights $w_i$
          ∗ Update $F_k(R) \leftarrow F_k(R) + \frac{1}{2} f_{k,l}(R)$, where $f_{k,l}$ is the best classifier among $\{f_{k,t}\}_{t=1...200}$ which minimizes the negative binomial log-likelihood (4.14) and $p_k(R) \leftarrow \frac{e^{F_k(R)}}{e^{F_k(R)} + e^{-F_k(R)}}$
          ∗ Sort positive and negative samples according to descending probabilities and find samples at the decision boundaries
            $R_p = (0.998N_p)$-th $R^+$, $R_n = (0.35N_n)$-th $R^-$
          ∗ $l = l + 1$
      – Store $F_k = \left\{(r_{k,l}, \boldsymbol{\mu}_{k,l}, g_{k,l})\right\}_{l=1...L_k}$ and $thrd_k = F_k(R_n)$
```

Figure 4.6: Pedestrian detection with cascade of LogitBoost classifiers on Sym_8^+.

are evaluated for most of the test samples, which produce a very efficient solution.

4.6 Experiments

We conduct experiments on two challenging datasets, INRIA and DaimlerChrysler. We compare the performance of our method with the best results published on the given datasets. In addition, we present several detection examples for crowded scenes.

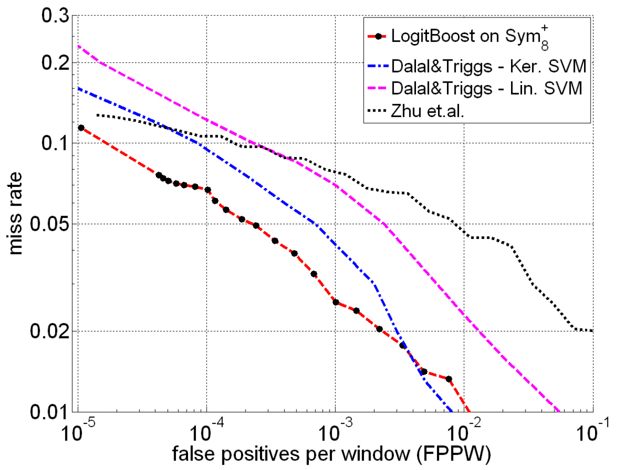


Figure 4.7: Comparison with methods of Dalal & Triggs [36] and Zhu et.al. [181] on INRIA dataset. The curves for other approaches are generated from the original papers.

4.6.1 Experiments on INRIA Dataset

INRIA pedestrian dataset [36] contains 1774 pedestrian annotations (3548 with reflections) and 1671 person free images. The pedestrian annotations were scaled into a fixed size of 64×128 windows which include a margin of 16 pixels around the pedestrians. The dataset was divided into two, where 2416 pedestrian annotations and 1218 person free images were selected as the training set, and 1132 pedestrian annotations and 453 person free images were selected as the test set. Detection on INRIA pedestrian dataset is challenging since it includes subjects with a wide range of variations in pose, clothing, illumination, background and partial occlusions.

In the *first experiment*, we compare our results with [36] and [181]. Although it has been noted that kernel SVM is computationally expensive, we consider both the linear and kernel SVM method of [36]. In [181], a cascade of AdaBoost classifiers was trained using HOG features, and two different results were reported based on the normalization of the descriptors. Here, we consider only the best performing result, the L_2-norm.

In Figure 4.7, we plot the detection error tradeoff curves on a log-log scale. The y-axis corresponds to the miss rate $\frac{\text{FalseNeg}}{\text{FalseNeg+TruePos}}$, and the x-axis corresponds to false positives per window (FPPW) $\frac{\text{FalsePos}}{\text{TrueNeg+FalsePos}}$. The curve for our method is generated by adding one cascade level at a time. For example, in our case the rightmost marker at $7.5 * 10^{-3}$ FPPW corresponds to detection using only the first 11 levels of cascade,

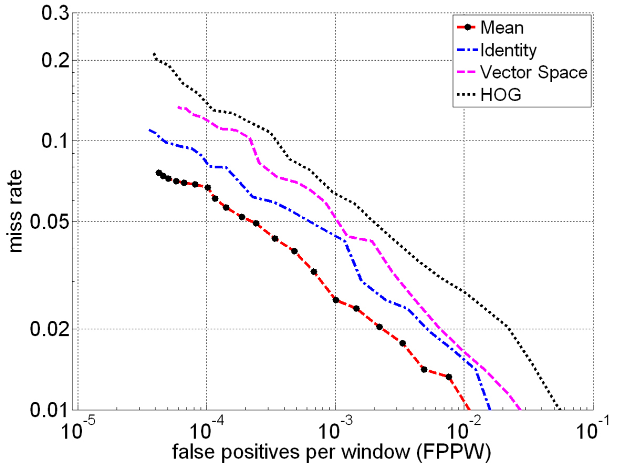
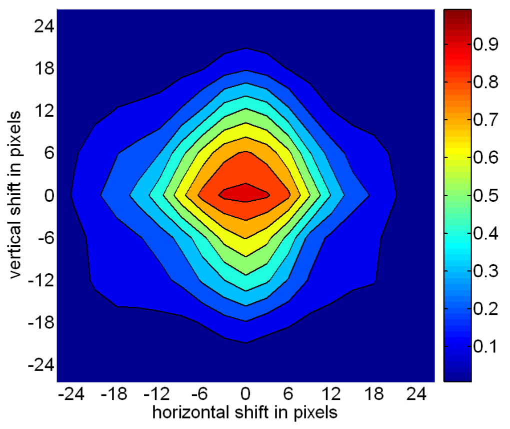


Figure 4.8: Detection rates of different approaches for our method on INRIA dataset. The charts at the weighted means significantly improve the results.

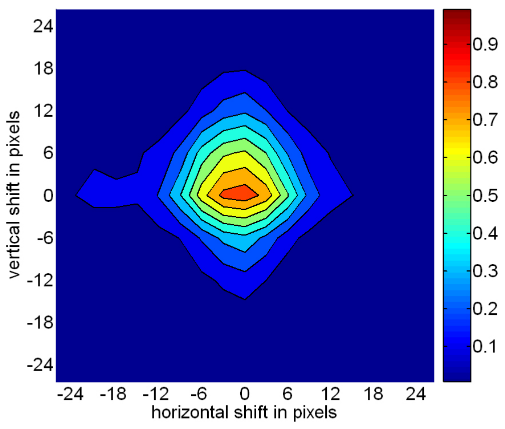
whereas the marker positioned at $4*10^{-5}$ FPPW corresponds to cascade of all 30 levels. The markers between the two extremes correspond to a cascade of between 11 to 30 levels.

To generate the result at 10^{-5} FPPW (leftmost marker), we shifted the decision boundaries of all the cascade levels, $thrd_k$, to produce less false positives at the cost of higher miss rates. We see that at almost all the false positive rates, our miss rates are significantly lower than the other approaches. The closest result to our method is the kernel SVM classifier of [36], which requires kernel evaluation at 1024 dimensional space to classify a single detection window. If we consider 10^{-4} as an acceptable FPPW, our miss rate is 6.8%, where the second best result is 9.3%.

Since the method removes samples which were rejected by the previous levels of cascade, during the training of last levels, only very small amount of negative samples, order of 10^2, remained. At these levels, the training error did not generalize well, such that the same detection rates are not achieved on the test set. This can be seen by the dense markers around FPPW $< 7 * 10^{-5}$. We believe that better detection rates can be achieved at low false positive rates with introduction of more negative images. In our method, 25% of false positives are originated from a single image which contains a flower texture, where the training set does not include a similar example.



(a)



(b)

Figure 4.9: Translation sensitivity. **(a)** Covariance and **(b)** HOG descriptors. Covariance descriptors have a larger region of support.

In the *second experiment*, we consider an empirical validation of the presented classification algorithm on Riemannian manifolds. In Figure 4.8, we present the detection error tradeoff curves for four different approaches.

- The original method, which maps the points to the tangent spaces at the weighted means.
- The mean computation step is removed from the original algorithm and points are always mapped to the tangent space at the identity.
- We ignore the geometry of Sym_8^+, and stack the upper triangular part of the covariance matrix into a vector, such that learning is performed on the vector space.
- We replace the covariance descriptors with HOG descriptors, and perform original (vector space) LogitBoost classification.

The original method outperforms all the other approaches significantly. The second best result is achieved by mapping points to the tangent space at the identity matrix followed by the vector space approaches. Notice that, our LogitBoost implementation utilizing HOG descriptors has 3% more miss rate at 10^{-4} FPPW than [181] which trains an AdaBoost classifier. The performance is significantly degraded beyond this point.

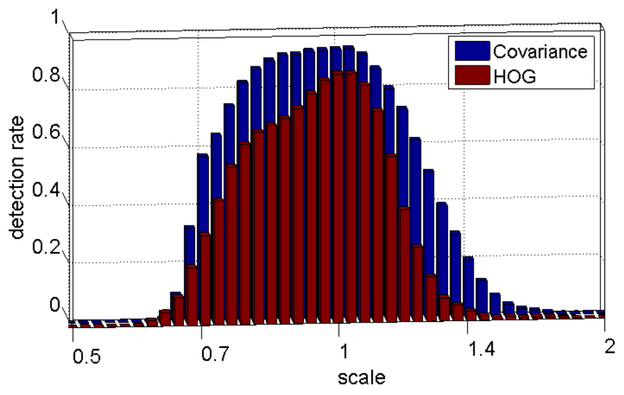


Figure 4.10: Scale sensitivity. The x-axis is plotted at log scale. Covariance descriptors are less sensitive to small scale changes.

In the *third experiment*, we examine the sensitivity of the covariance and HOG descriptors to translation and scaling of the target windows relative to the original position. The performance of the HOG descriptors is tested with our implementation. The false positive rates of both classifiers are fixed at 10^{-4} FPPW. The translation sensitivity is presented in Figure 4.9, where we observe that the covariance descriptors are less sensitive to small translations of the target windows. The detection rate is almost constant for ± 6 pixels translation which approximately corresponds to 10% translation in x-axis and 5% translation in y-axis of the target window (64×128).

The scale sensitivity of both methods are presented in Figure 4.10, where we again observe that the covariance descriptors are less sensitive to small scalings of the target windows. For covariance descriptor, the detection rates are almost constant for $\pm 20\%$ scalings, and gradually decrease beyond.

For detection applications, there is a tradeoff between low and high sensitivity of the detector with respect to small transformations. While detecting objects in novel scenes, objects are searched through the transformation space. A detector with invariance to small transformations have the advantage of reducing the size of the search space, whereas the space should be searched more densely with a high sensitivity detector. Besides, a less sensitive detector is more desirable when the target objects have high variability and training data is not perfectly aligned. On the other hand, a highly sensitive detector can better localize the targets. In Figures 4.9 and 4.10, we see that

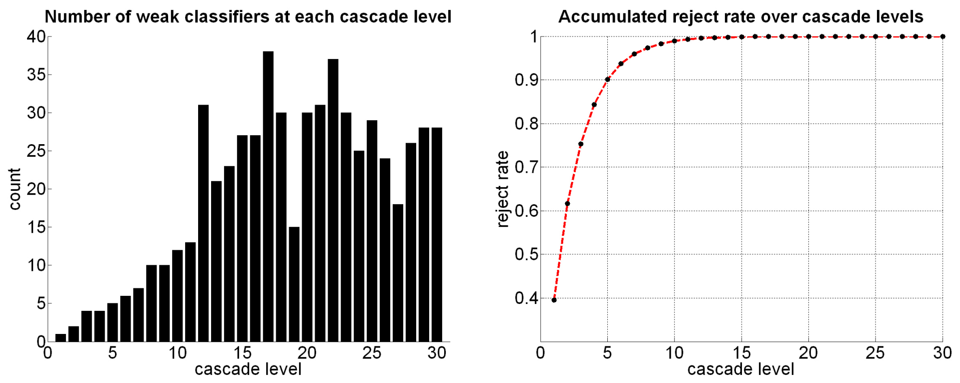


Figure 4.11: The number of weak classifiers at each cascade level and the accumulated rejection rate over the cascade levels. On average our method requires evaluation of 8.45 covariance descriptors per negative detection window.

the detection rates for our approach are smooth and symmetric functions with respective to both translation and scale having peak at the original location of the target. Therefore, with a simple maxima search we can accurately localize the targets. Please see Section 4.6.3 for more details.

In Figure 4.11, we plot the number of weak classifiers at each cascade level and the accumulated rejection rate over the cascade levels. There are very few classifiers on early levels of cascade and the first five levels reject 90% of the negative examples. On average our method requires evaluation of 8.45 covariance descriptors per negative detection window, whereas on average 15.62 HOG evaluations were required in [181].

4.6.2 Experiments on DaimlerChrysler Dataset

DaimlerChrysler dataset [115] contains 4000 pedestrian (24000 with reflections and small shifts) and 25000 non-pedestrian annotations. As opposed to INRIA dataset, non-pedestrian annotations were selected by a preprocessing step from the negative samples, which match a pedestrian shape template based on average Chamfer distance score. Both annotations were scaled into a fixed size of 18×36 windows, and pedestrian annotations include a margin of 2 pixels around. The dataset was organized into three training and two test sets, each of them having 4800 positive and 5000 negative examples. The small size of the windows combined with a carefully arranged negative set makes detection on DaimlerChrysler dataset extremely challenging. In addition,

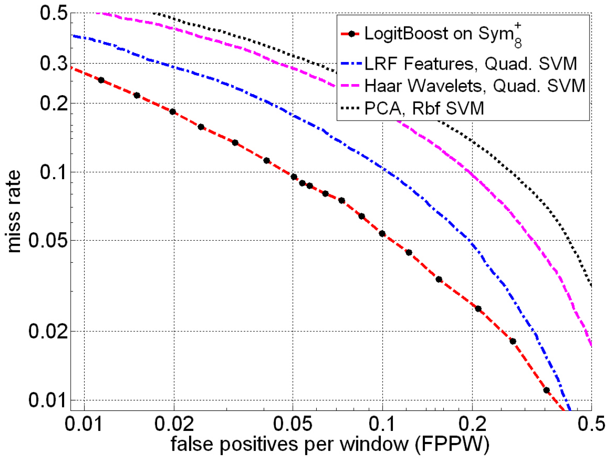


Figure 4.12: Comparison with [115] on DaimlerChrysler dataset. The curves for other approaches are generated from the original papers.

3600 person free images with varying sizes between 360×288 and 640×480 were also supplied.

In [115], an experimental study was described comparing three different feature descriptors and various classification techniques. The compared feature descriptors were the PCA coefficients, Haar Wavelets and local receptive fields (LRFs) which are the output of the hidden layer of a specially designed feed forward NN. The connections of the neurons in the hidden layer of the NN were restricted to local regions of the image, and the hidden layers were divided into branches with all the neurons sharing the same set of weights. Although several other classification methods were also considered, the best detection performances among all the different features were achieved utilizing SVMs.

In the *first experiment*, we compare our method with the best results for each descriptor in [115]. The same training configuration is prepared by selecting two out of three training sets. Since the number of non-pedestrian annotations was very limited for training of our method, we adapted the training parameters. A cascade of $K = 15$ LogitBoost classifiers on Sym_8^+ is learned, where each level is optimized to detect at least 99.75% of the positive examples, while rejecting at least 25% negative samples.

In Figure 4.12, we plot the detection error tradeoff curves. The cascade of 15 LogitBoost classifiers produced a FPPW rate of 0.05. The detection rates with lower FPPW

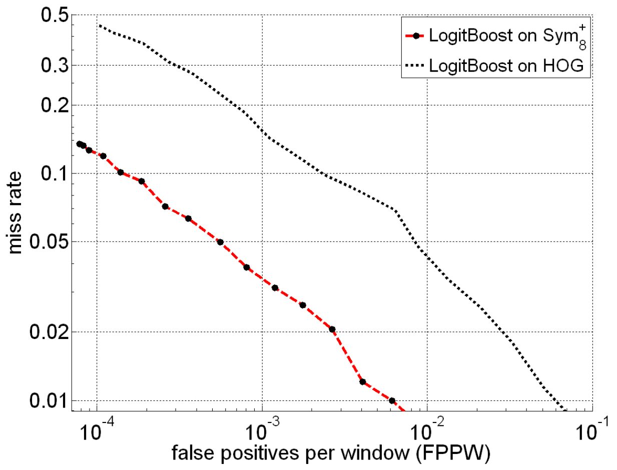


Figure 4.13: Comparison of covariance and HOG descriptors on DaimlerChrysler dataset.

are generated by shifting the decision boundaries of all the cascade levels gradually, until FPPW $=$ 0.01. We see that our approach has significantly lower miss rates at all the false positive rates. This experiment should not be confused with the experiments on INRIA dataset, where much lower FPPW rates were observed. Here, the negative set consists of hard examples selected by a preprocessing step.

In the *second experiment*, we set up a different test configuration on DaimlerChrysler dataset. The 3600 person free images are divided into two, where 2400 images are selected as the negative training set, and 1200 images are selected for the negative test set. For both the covariance descriptors and the HOG descriptors, we trained cascade of $K = 25$ classifiers. We observed that the object sizes were too small for HOG descriptors to separate among positive and negative examples at the later levels of cascade. The classifiers trained utilizing HOG descriptors failed to achieve the specified detection (99.8%) and the rejection rates (35.0%). We stopped adding weak learners to a cascade level after reaching $L_k = 100$. The detection error tradeoff curves are given in Figure 4.13 where we see that the covariance descriptors significantly outperform HOG descriptors.

4.6.3 Detection Examples

Since the sizes of the pedestrians in novel scenes are not known apriori, the images are searched at multiple scales. There are two searching strategies. The first strategy is to scale the detection window and apply the classifier at multiple scales. The second strategy is to scale the image and apply the classifier at the original scale. In covariance representation we utilized gradient based features which are scale dependent. Therefore evaluating classifier at the original scale (second strategy) produces the optimal result. However, in practice up to scales of 2x we observed that the detection rates were almost the same, whereas in more extreme scale changes the performance of the first strategy degraded. The drawback of the second strategy is slightly increased search time, since the method requires computation of the filters and the integral representation at multiple scales.

Utilizing the classifier trained on the INRIA dataset, we generated several detection examples for crowded scenes with pedestrians having variable illumination, appearance, pose and partial occlusion. The results are shown in Figure 4.14. The images are searched at five scales using the first strategy, starting with the original window size 64×128 and two smaller and two larger scales of ratio 1.2. The white dots are all the detection results and we filtered them with adaptive bandwidth mean shift filtering [27] with bandwidth 1/10 of the window width and height. Black dots show the modes, and ellipses are generated by averaging the detection window sizes converging to the modes.

4.6.4 Computational Complexity

The training of the classifiers took two days on a Pentium D 2.80Ghz processor with 2.00GB of RAM with a C++ implementation, which is a reasonable time to train a cascade model. Computation of tensor of integral images requires $\mathcal{O}(d^2WH)$ arithmetic operations, which approximately takes 0.1 seconds for a 320×240 image. The computation of the normalized covariance descriptor of an arbitrary region requires $\mathcal{O}(d^2)$ operations using the integral structures and is invariant of the region size.

The most computationally expensive operation during the classification is the eigenvalue decomposition to compute the logarithm of a symmetric matrix, which requires $\mathcal{O}(d^3)$ arithmetic operations. Given a test image, on average the method can search around 3000 detection windows per second, which approximately corresponds to 3 seconds for a dense scan of a 320×240 image, 3 pixel jumps vertically and horizontally.

The proposed learning algorithm is not specific to Sym_d^+, and can be used to train classifiers for points lying on any connected Riemannian manifold. In addition, the approach can be combined with any boosting method. During our experiments, we implemented LogitBoost, GentleBoost and AdaBoost classifiers on Riemannian manifolds using LDA, decision stumps and linear SVMs as weak learners. The results of the methods were comparative. Due to simplicity (training time and ease of implementation) and slightly better performance, we presented the LogitBoost algorithm.

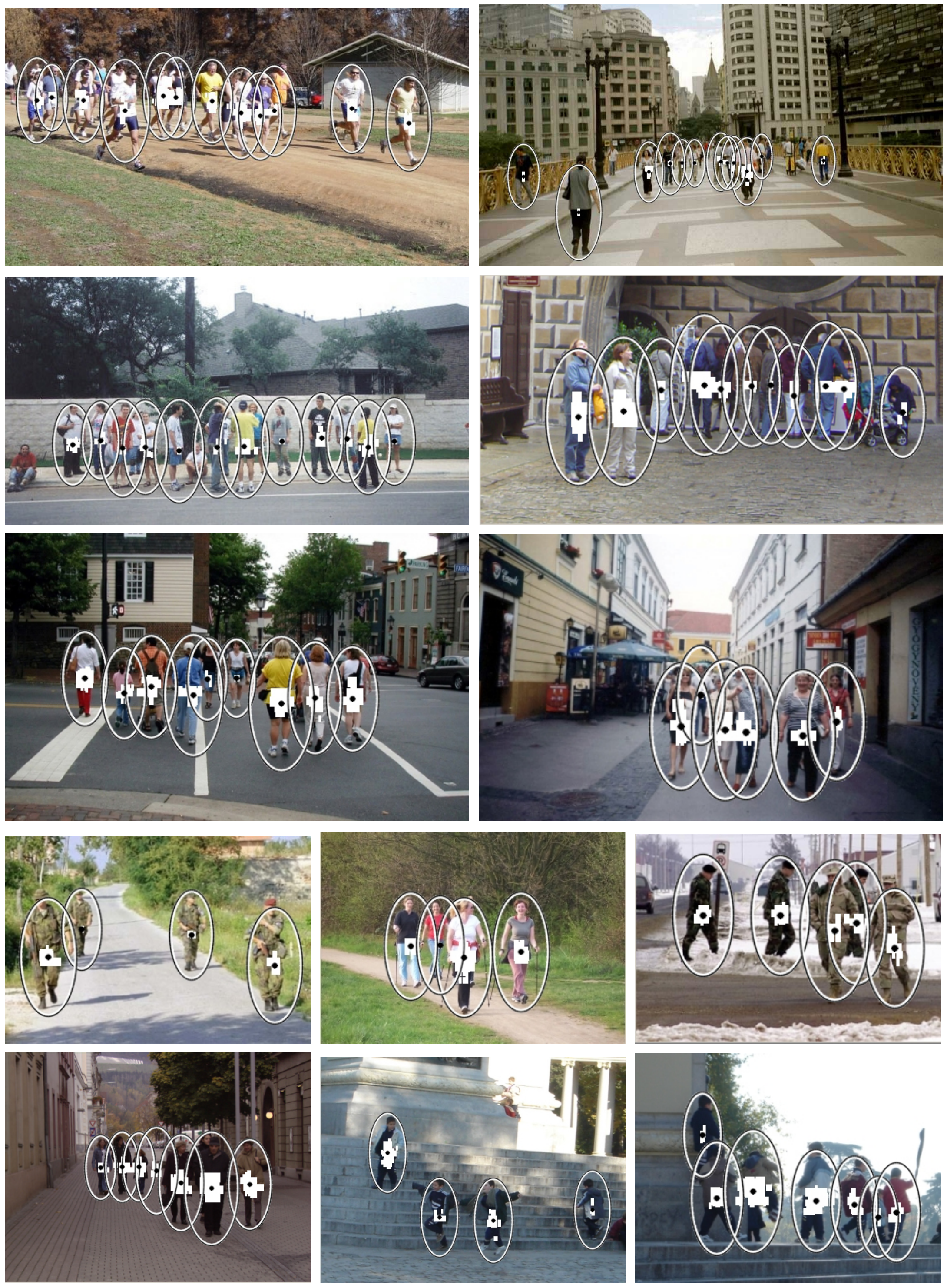


Figure 4.14: Detection examples. The classifier is trained on the INRIA dataset. White dots show all the detection results. Black dots are the modes generated by mean shift smoothing and the ellipses are average detection window sizes. There are extremely few false positives and negatives.

Chapter 5

Regression on Lie Group Transformations

5.1 Introduction

Regression analysis is a statistical technique to recover the relationship between one or more predictor (independent) variables and a response (dependent) variable based on the observed data. The analysis can be extended to multiple response setting by assessing the dependence of multi dimensional response variables to predictor variables. Several important computer vision problems can be formulated as a multiple response regression problem. For instance, tracking or motion estimation can be considered as learning the dependence of motion parameters to the image measurements.

The classical regression techniques can be generalized to multiple response setting by learning the relationships between the predictor variables and each of the response variables independently. However, these methods fail to model the dependence among the response variables. A more difficult situation appears if the response variables do not form a vector space. For instance, assume that we want to learn a regression function which predicts affine motions based on image observations. If each of the response parameters is estimated independently, the regressor can output a singular matrix which is not contained in the space. To solve the problem an additional parametrization of the response space is required. Secondly, the geometry of the underlying space should be considered while learning the regressors, such as the minimized error function.

Here we present a novel approach for learning a regression function where the range of the regressors form a matrix Lie group. Without loss of generality we describe the approach on the space of affine matrices, however the method generalizes to any matrix Lie group. We present an application of the derived learning algorithm for the affine tracking problem and its extension to invariant object detection [156].

An overview of the existing tracking techniques was presented in Section 3.5. Here we reference only the template alignment methods which are most relevant to our application. Template alignment methods define a cost function based on the difference between the object template and the image measurements, and solve an optimization problem on the motion parameters. The most famous example is the optical flow estimation [101], where the sum of squared difference between the template and the image intensities was minimized as an iterative least squares problem. Since the method requires computation of the image gradient, the Jacobian and the Hessian for each iteration, it is computationally intensive. Several variants of the method were proposed [64, 143] to overcome the difficulty and a comprehensive overview of these models can be found in [105].

In [30, 86], the motion was estimated using a linear function of the image gradient, which was learned in an off-line process. Later [171], the idea was extended to learn a nonlinear model using relevance vector machine. Notice that, these methods estimate the additive updates to the motion parameters via linearization.

There are only a few references which explicitly consider the geometry of the response variables for learning based approaches. In [39], an addition operation was defined on the Lie algebra for tracking an affine snake. In [7], the additive updates were performed on the Lie algebra for template tracking. However, the approach in [7] fails to account for the noncommutativity of the matrix multiplications and the estimations become valid only around the initial transformation of the target. In a recent study [37], a kernel regression model for manifold valued data is described for analyzing shape changes of the brain on MR images. The approach is computationally expensive and is not well suited for real time applications such as tracking.

Since the existing techniques for template alignment proceed by linearizing the motion, an implicit Euclidean space assumption is made. Several transformations used in computer vision have matrix Lie group structure. We present a novel formulation for motion estimation by learning a regression model on the Lie algebra and show that the formulation minimizes a first order approximation to the geodesic error. The appearance of an object template is described with several orientation histograms computed

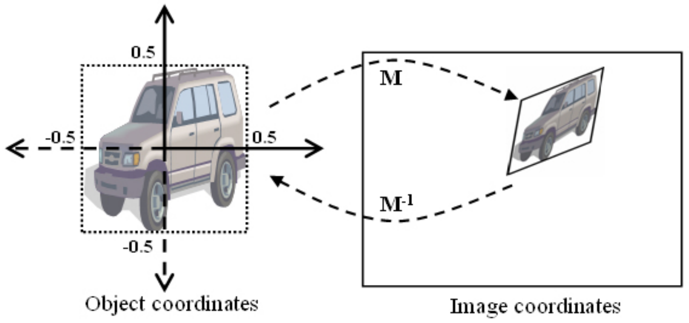


Figure 5.1: The mapping and its inverse, between the object and image coordinates.

on a regular grid. Using a regression function, we learn the dependence of the motion to the observed descriptors.

Majority of the current state of art object detection algorithms are based on sequentially applying a learned classifier of the object model at all the possible subwindows. However, a brute force approach on a high dimensional search space is computationally intractable. The proposed learning model is extended to train a class specific tracking function which can localize the targets with a sparse scan on the motion space. The motion estimator is then integrated to an existing pose dependent object detector and a pose invariant object detection algorithm with respect to the motion model is developed.

5.2 Tracking as a Learning Problem

Without loss of generality, the method is demonstrated on affine motions, however, it generalizes to any matrix Lie group transformations.

A two-dimensional affine transformation is given by a 3×3 matrix $\mathbf{M}$

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & 1 \end{pmatrix} \tag{5.1}$$

where $\mathbf{A}$ is a nonsingular 2×2 matrix and $\mathbf{b} \in \mathbb{R}^2$. Let $\mathbf{M}$ transforms a unit square at the origin to the affine region enclosing the target object

$$[x_{img} \quad y_{img} \quad 1]^T = \mathbf{M}[x_{obj} \quad y_{obj} \quad 1]^T \tag{5.2}$$

where, the subscripts indicate the object coordinates and image coordinates respectively. The inverse $\mathbf{M}^{-1}$ is also an affine motion matrix and transforms the image coordinates to the object coordinates (Figure 5.1).

Let I denote the observed images and t be the time index. The aim of tracking is to estimate the transformation matrix $\mathbf{M}_t$, given the observations up to time t, $I_{0\ldots t}$, and the initial transformation $\mathbf{M}_0$. We model the transformations incrementally

$$\mathbf{M}_t = \mathbf{M}_{t-1}.\Delta\mathbf{M}_t \tag{5.3}$$

and estimate the increments $\Delta\mathbf{M}_t$ at each time frame. The transformation $\Delta\mathbf{M}_t$ corresponds to motion of target from time $t-1$ to t in the object coordinates.

The image in the object coordinates is written as $I(\mathbf{M}^{-1})$. We consider the pixel values inside the unit rectangle and represent the region with a descriptor, such as, orientation histograms. It is denoted by $\mathbf{o}(\mathbf{M}^{-1}) \in \mathbb{R}^n$ where n is the dimension of the descriptor.

We interpret tracking as a matrix valued regression problem. Given the previous location of the object $\mathbf{M}_{t-1}$ and the current observation I_t, we estimate the new transformation $\Delta\mathbf{M}_t$ by the regression function

$$\Delta\mathbf{M}_t = f(\mathbf{o}_t(\mathbf{M}_{t-1}^{-1})). \tag{5.4}$$

The problem reduces to learning and updating the regression function f, where the details are explained in Section 5.4.

During initialization, $t = 0$, the observation I_0 and the initial location of the object $\mathbf{M}_0$ are given. We generate a training set of N random affine transformation matrices $\{\Delta\mathbf{M}\}_{i=1\ldots N}$ around the identity matrix. The object coordinates are transformed by multiplying on the left with $\Delta\mathbf{M}_i^{-1}$ and the new descriptor is computed by $\mathbf{o}_0^i = \mathbf{o}_0\left(\Delta\mathbf{M}_i^{-1}.\mathbf{M}_0^{-1}\right)$. The transformation $\Delta\mathbf{M}_i$ moves the object back to the unit square. The training set consists of samples $\left\{\mathbf{o}_0^i, \Delta\mathbf{M}_i\right\}_{i=1\ldots N}$. The process is illustrated in Figure 5.2. Notice that we use the notation $\Delta\mathbf{M}$ both for the elements of the training set with the subscript i, and the estimated motions during tracking with the subscript t.

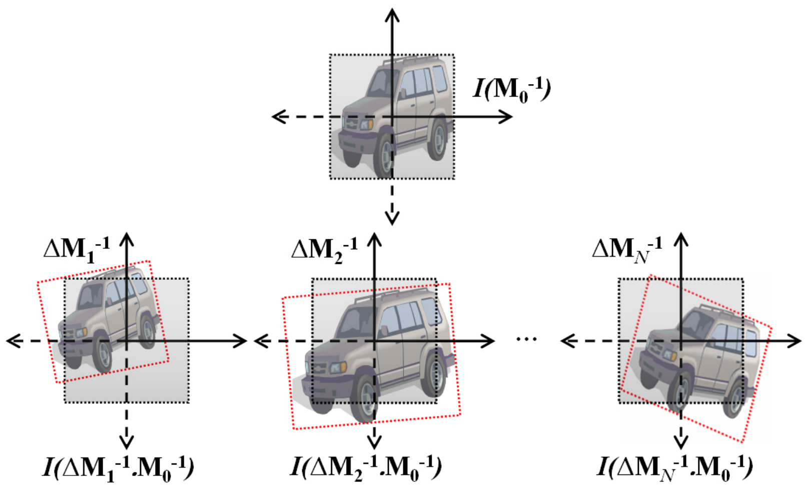

$I(\mathbf{M}_0^{-1})$
$\Delta\mathbf{M}_1^{-1}$
$\Delta\mathbf{M}_2^{-1}$
$\Delta\mathbf{M}_N^{-1}$
...
$I(\Delta\mathbf{M}_1^{-1}.\mathbf{M}_0^{-1})$
$I(\Delta\mathbf{M}_2^{-1}.\mathbf{M}_0^{-1})$
$I(\Delta\mathbf{M}_N^{-1}.\mathbf{M}_0^{-1})$

Figure 5.2: Training samples are generated by applying N affine motions $\Delta\mathbf{M}_{i=1\ldots N}^{-1}$ at the object coordinates. Using (5.3), the new observations in the object coordinates are $I_0\left((\mathbf{M}_0.\Delta\mathbf{M}_i)^{-1}\right)$, where an equivalent form is used in the image.

The regression function $f : \mathbb{R}^n \mapsto A(2)$ is an affine matrix valued function. The standard approach for motion estimation is through introducing a parametrization of the motion and linearization [30, 86, 171], which in this case is around the identity matrix

$$\Delta\mathbf{M}(\mathbf{p_0} + \Delta\mathbf{p}) \approx \Delta\mathbf{M}(\mathbf{p}_0) + \frac{\partial \Delta\mathbf{M}}{\partial \mathbf{p}}\Delta\mathbf{p}. \tag{5.5}$$

where $\Delta\mathbf{M}(\mathbf{p_0}) = \mathbf{I}$. The approach proceeds by estimating the increments $\Delta\mathbf{p}$. There are two major drawbacks of the approach. Firstly, the approximation makes a Euclidean space assumption on the parameters. Secondly, the parametrization is arbitrary and do not consider the structure of the motion. We use the Lie group theory to estimate the tracking function.

5.3 Affine Group

The structure of affine matrices was given in (5.1). The set of all two-dimensional affine transformations form a matrix Lie group denoted by $A(2)$. The associated Lie algebra is the set of matrices

$$\mathbf{m} = \begin{pmatrix} \mathbf{U} & \mathbf{v} \\ 0 & 0 \end{pmatrix} \tag{5.6}$$

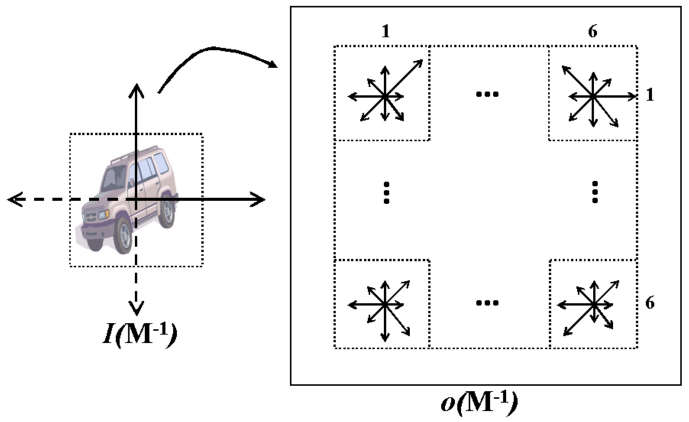


Figure 5.3: The gradient weighted orientation histograms are utilized as region descriptors.

where, $\mathbf{U}$ is a 2×2 matrix and $\mathbf{v} \in \mathbb{R}^2$. The matrix $\mathbf{m}$ is sometimes referred to as a $m = 6$ dimensional vector by selecting each of the entries of $\mathbf{U}$ and $\mathbf{v}$ as an orthonormal basis. Please see Section A.7.1 for more details on Lie groups.

5.4 Tracking via Regression on Lie Groups

In this section, we present the learning and the update of the tracking function.

5.4.1 Region Descriptor

The target region is represented with several orientation histograms [53] computed at a regular grid inside the unit square in object coordinates (Figure 5.3). Similar to SIFT descriptors [100], the contribution of each pixel to the histogram is proportional to its gradient magnitude. The unit square is divided into $6 \times 6 = 36$ regions and a histogram is computed in each of them. Each histogram is quantized at $\pi/4$ degrees between 0 and 2π. The size of each histogram is eight dimensional and the descriptors, $\mathbf{o}$, are $n = 288$ dimensional. During tracking the peripheral pixels are frequently contaminated by the background, hence we leave a 10% boundary at the outerside of the unit square and construct the descriptor inside the inner rectangle.

5.4.2 Model Learning

During the model learning, the parameters of the regression function, $f : \mathbb{R}^n \mapsto A(2)$, are estimated. The training set consists of samples $\left\{\mathbf{o}_0^i, \Delta\mathbf{M}_i\right\}_{i=1\ldots N}$. The affine motion matrices do not form a vector space therefore the Euclidean distance between two motions is not an appropriate metric. Based on the geometry of the space a meaningful error function is the sum of the squared geodesic distances between the estimations $f(\mathbf{o}_0^i)$, and the true transformations $\Delta\mathbf{M}_i$

$$J_g = \sum_{i=1}^{N} d^2\left[f(\mathbf{o}_0^i), \Delta\mathbf{M}_i\right] \tag{5.7}$$

where the distance, d, is given in (A.20).

Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be two motion matrices, and let $\mathbf{m}_1 = \log(\mathbf{M}_1)$ and $\mathbf{m}_2 = \log(\mathbf{M}_2)$. Using Baker-Campbell-Hausdorff formula (A.16) which gives the exponential identity for non-commutative Lie groups, a first order approximation to the geodesic distance between the two motion matrices is given by

$$\begin{aligned} d(\mathbf{M}_1, \mathbf{M}_2) &= \left\|\log\left[\mathbf{M}_1^{-1}\mathbf{M}_2\right]\right\| \\ &= \left\|\log\left[\exp(-\mathbf{m}_1)\exp(\mathbf{m}_2)\right]\right\| \\ &= \left\|\log\left[\exp(\mathbf{m}_2 - \mathbf{m}_1 + \mathcal{O}(|(\mathbf{m}_1, \mathbf{m}_2)|^2)\right]\right\| \\ &\approx \left\|\mathbf{m}_2 - \mathbf{m}_1\right\|. \end{aligned} \tag{5.8}$$

Selecting m orthonormal bases on the Lie algebra, we can compute the matrix norm as the Euclidean distance between two vectors. Using (5.8), the function (5.7) is equivalent to minimizing

$$J_a = \sum_{i=1}^{N} \left\|\log\left(f(\mathbf{o}_0^i)\right) - \log\left(\Delta\mathbf{M}_i\right)\right\|^2 \tag{5.9}$$

up to first order terms. The approximation is good enough since the transformations are in a small neighborhood of the identity.

We define the regression function as

$$f(\mathbf{o}) = \exp\left(g(\mathbf{o})\right) \tag{5.10}$$

and learn the function $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ which estimates the tangent vectors, $\log(\Delta\mathbf{M})$, on the Lie algebra. We model the function g as a linear function of the observations $\mathbf{o}$

$$g(\mathbf{o}) = \mathbf{o}^T \mathbf{\Omega} \tag{5.11}$$

where $\mathbf{\Omega}$ is the $n \times m$ matrix of regression coefficients.

Let $\mathbf{X}$ be the $N \times n$ matrix of initial observations and $\mathbf{Y}$ be the $N \times m$ matrix of mappings of motions to the Lie algebra

$$\mathbf{X} = \begin{pmatrix} \left[\mathbf{o}_0^1\right]^T \\ \vdots \\ \left[\mathbf{o}_0^N\right]^T \end{pmatrix} \qquad \mathbf{Y} = \begin{pmatrix} [\log(\Delta\mathbf{M}_1)]^T \\ \vdots \\ [\log(\Delta\mathbf{M}_N)]^T \end{pmatrix}. \tag{5.12}$$

Notice that, $\log(\Delta\mathbf{M})$ is referred here in m-dimensional vector form. Substituting (5.10) and (5.11) into (5.9), we obtain

$$J_a = tr[(\mathbf{X}\mathbf{\Omega} - \mathbf{Y})^T(\mathbf{X}\mathbf{\Omega} - \mathbf{Y})] \tag{5.13}$$

where the trace replaces the summation in (5.9).

For real time tracking we keep the size of the training set relatively small, $N = 200$. Since number of samples is smaller than the dimension of the feature space, $N < n$, the system is underdetermined and the least squares estimate becomes inaccurate. To avoid overfitting, we introduce an additional constraint on the size of the regression coefficients

$$J_r = tr[(\mathbf{X}\mathbf{\Omega} - \mathbf{Y})^T(\mathbf{X}\mathbf{\Omega} - \mathbf{Y})] + \lambda\|\mathbf{\Omega}\|^2 \tag{5.14}$$

which is called the *ridge regression* [71, p.59-64]. The minimum of the error function J_r is given by

$$\mathbf{\Omega} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.15}$$

where $\mathbf{I}$ is $n \times n$ identity matrix. The regularization coefficient λ determines the degree of shrinkage on the regression coefficients. The details of the parameter λ is explained in Section 5.6.

```
Input: Location of target at time $t-1$ is $\mathbf{M}_{t-1}$ and the current observation is $I_t$, maximum iteration number is $K$.
    • $k = 1$ and $\mathbf{M}_t = \mathbf{M}_{t-1}$
    • Repeat
        – $\Delta\mathbf{M}_t = f(\mathbf{o}_t(\mathbf{M}_t^{-1}))$
        – $\mathbf{M}_t = \mathbf{M}_t.\Delta\mathbf{M}_t$
        – $k = k + 1$
    • Until $\Delta\mathbf{M}_t = \mathbf{I}$ or $k = K$
```

Figure 5.4: Tracking algorithm.

5.4.3 Interframe Correspondence

After learning the regression function f, the tracking problem reduces to estimating the motion via (5.4) using current observation I_t and updating the target location via (5.3). To better localize the target, in each frame we repeat the motion estimation using f a maximum of ten times or $\Delta\mathbf{M}_t$ becomes equal to identity (Figure 5.4).

5.4.4 Model Update

Since objects can undergo appearance changes in time, it is necessary to adapt to these variations. In our case, the model update reestimates the tracking function f. During tracking the target object, we generate $s = 2$ random observations at each frame with the same method described in Section 5.2. The observations stored for last $p = 100$ frames constitute the update training set. Let $\mathbf{X}_u$ and $\mathbf{Y}_u$ be the update training set stored in the matrix form as described in (5.12), and $\mathbf{\Omega}'$ be the previous model parameters. After each p frames of tracking, we update the coefficients of the regression function by minimizing the error function

$$\begin{aligned} J_u = tr[(\mathbf{X}_u\mathbf{\Omega} - \mathbf{Y}_u)^T(\mathbf{X}_u\mathbf{\Omega} - \mathbf{Y}_u)] + \\ \lambda\|\mathbf{\Omega}\|^2 + \gamma\|\mathbf{\Omega} - \mathbf{\Omega}'\|^2. \end{aligned} \tag{5.16}$$

The error function is similar to (5.14), but another constraint is introduced on the difference of regression coefficients. Differentiating the error function J_u with respect to $\mathbf{\Omega}$, the minimum is achieved at

$$\mathbf{\Omega} = (\mathbf{X}_u^T \mathbf{X}_u + (\lambda + \gamma)\mathbf{I})^{-1}(\mathbf{X}_u^T \mathbf{Y}_u + \gamma \mathbf{\Omega}'). \tag{5.17}$$

The parameter γ controls how much change on the regression parameters are allowed from the last estimation. More details of the parameter γ is explained in Section 5.6. To take into account the bias terms all the function estimations are performed using centered data.

5.5 Invariant Object Detection

In [180], it was argued that scanning of the whole image for detecting anatomic structures in medical images is unnecessary since the problem domain offers strong contextual information for localizing the targets. Utilizing a similar idea, we present a method to build an invariant detection algorithm by integrating a class specific tracking function to an existing pose dependent detector. We demonstrate the approach for affine invariant detection of faces.

We perform a sparse scan of the image, and determine all the possible object locations with a pre-learned class specific tracking function (e.g. tracker for faces). The tracker finds all the locations in the motion space (e.g. affine) which resemble the object model. The object detector is then evaluated only at these locations.

The benefits of the approach is two-fold. Firstly, the size of the search space drastically reduces. For example, we only consider a tracker which can correctly estimate translational motions upto 1/4 of the object size. Then it is possible to scan the image with jumps equal to 1/2. The ratio of number of search locations compared to the brute force approach decreases exponentially with the dimensionality of the motion model. Secondly, the proposed method performs continuous estimation of the target pose, whereas the existing techniques perform search on a quantized space, e.g. rotations of $\pi/6$. Utilizing a pose dependent object detection algorithm (e.g., frontal faces in upright position), the method enables to detect objects in arbitrary poses.

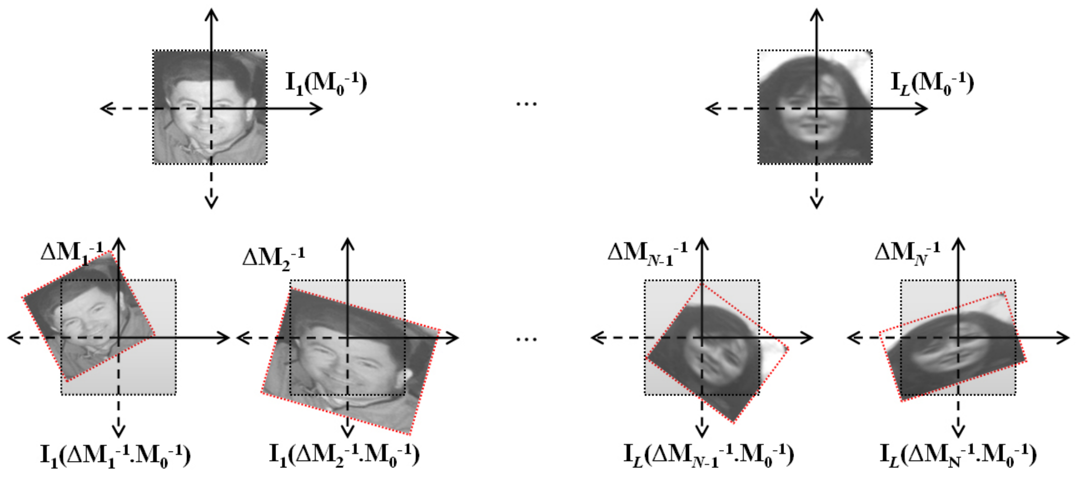


Figure 5.5: Training samples are generated by applying n affine motions $\Delta\mathbf{M}^{-1}_{i=1\ldots N}$ to L face images in the dataset.

5.5.1 Learning Class Model

Instead of learning a tracking function of the specific target object (5.4), we train a regression function of the object class. For instance, we consider a face tracker. The learning is performed on the training set generated by applying a total of N random affine transformations to L face images (Figure 5.5).

The training is an offline process and a more complicated model can be learned compared to tracking applications. However, the learned function should be evaluated fast at runtime, since the tracker is initiated at several locations for each test image. We consider two models for learning. First model is the ridge regression which was explained in Section 5.4.

As the second model, we consider the regression forest which is a bagged model of tree regressors [71, p.266-270]. Given a training set, we learn a binary tree model where each leaf node is a motion vector on the Lie algebra. The inner nodes make binary comparisons of two feature dimensions of the descriptor (out of 288), and based on the result split the space into two. During learning, 100 randomly selected features are evaluated at each node and the best pair which minimizes the sum of squared approximation error (5.9) on the divided spaces is selected. The growing of the tree at a node ends if there is a single sample inside or the depth of the tree reaches 15. The

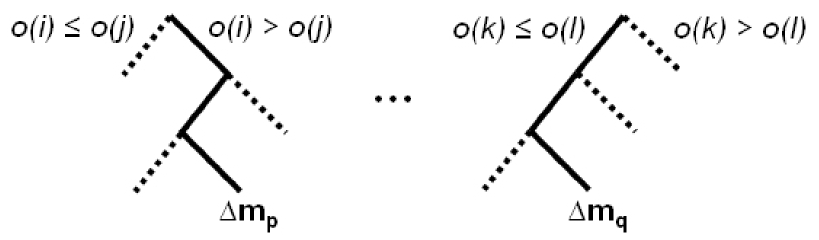


Figure 5.6: Regression forest.

value of the leaf nodes with multiple samples (the nodes with depth 15) are assigned to the mean. In general, a single tree model performs poorly since the variance of the model is very high. To reduce the variance, a bagged model is learned which consists of 100 binary tree regressors. Each of them is trained on a different training set of randomly generated motions. The estimation of the random forest is the average of the 100 motions estimated by the regression trees. The tree model is evaluated fast in real time since each tree performs an estimation by at most 15 binary comparisons. Regression forrest model is shown in Figure 5.6.

5.5.2 Detection

To detect faces in a given test image the trackers are initialized at sparse set of locations, on a regular grid with 1/2 jumps of the window size (minimum 24×24) and scales of factor 2, until image size. Each tracker is iterated $K = 20$ times and the final locations are evaluated with Viola and Jones face detector [165] (Figure 5.7).

5.6 Experiments

We present several experiments both on affine tracking and object detection.

5.6.1 Affine Tracking

In the *first experiment*, we compare the Lie algebra based parametrization with the linearization (5.5) around the identity matrix [30, 86, 171] by measuring the estimation errors. We also compare orientation histograms with the intensity difference features used in optical flow estimation and tracking [7, 30, 86].

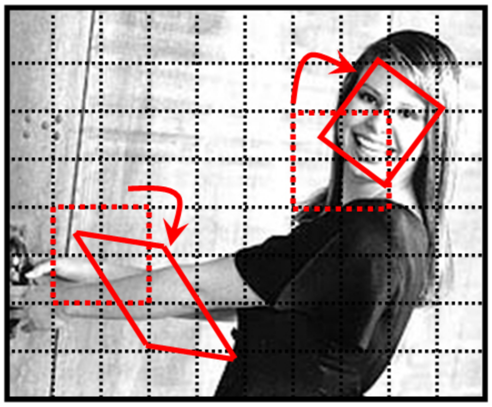


Figure 5.7: The trackers are initialized at sparse regular grid (dashed boxes). Final locations (solid boxes) are evaluated with face detector and the nonface regions are rejected.

We generated a training set of $N_{tr} = 200$ samples by random affine transformations of a single object. The motions are generated on the Lie algebra, by giving random values between -0.2 and 0.2 to each of the six parameters, and mapped to affine matrices via exponentiation. The function f is estimated by ridge regression with $\lambda = 2.10^{-3}$ for orientation histograms and $\lambda = 5.0$ for intensity features, determined by cross validation.

Each test set consists of $N_{te} = 1000$ samples. The samples inside a set have fixed norm. The norms $\|\log(\Delta\mathbf{M})\|$ vary from 0.025 to 0.35. We perform a single tracking iteration by each method, and measure the mean squared geodesic error (MSGE)

$$\frac{1}{N_{te}} \sum_{j=1}^{N_{te}} d^2 \left[f(\mathbf{o}_0^j), \Delta\mathbf{M}_j \right] \tag{5.18}$$

between the estimations and the true values (Figure 5.8). The estimation based on the Lie algebra is better than the linearization for transformation of all norms. The ratio is almost constant and on the average the linearization have 12% larger error. The estimations with orientation histograms are significantly better than the intensity based features.

In the *second experiment*, we show tracking examples for several challenging sequences. In *all* the experiments, the parameters of the ridge regression were $\lambda = \gamma = 2.10^{-3}$, which were learned offline via cross validation. The training dataset is generated on the Lie algebra, by giving random values between -0.1 and 0.1 to each of

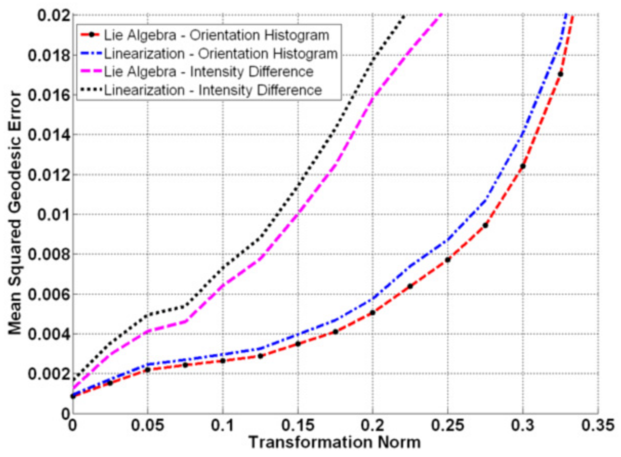


Figure 5.8: Estimation errors of the Lie algebra and the linearization methods using orientation histograms and intensity features.

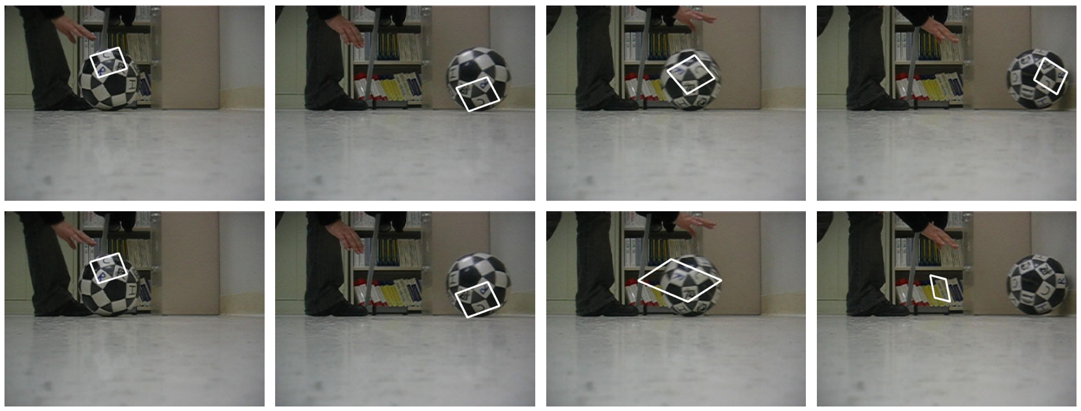


Figure 5.9: Comparison of the Lie algebra (*first row*) and the linearization (*second row*) based estimations. The target has large motions and the linearization based estimation loses the target after a few seconds. The sequence contains 318 frames.

the six parameters. Notice that, although we track the targets with an affine model, almost none of the targets are planar. Therefore, an affine model can not perfectly fit the target but produces the best affine approximation.

In Figure 5.9, a ball having large motions is tracked. The Lie algebra (first row) based estimation accurately tracks the target, whereas using linearization (second row) the target is lost after a few seconds. In the following sequences, only Lie algebra based estimations are shown.

Since nonplanar objects undergo significant appearance variations due to pose changes, the model update becomes important. In Figure 5.10, we show the effect of the update

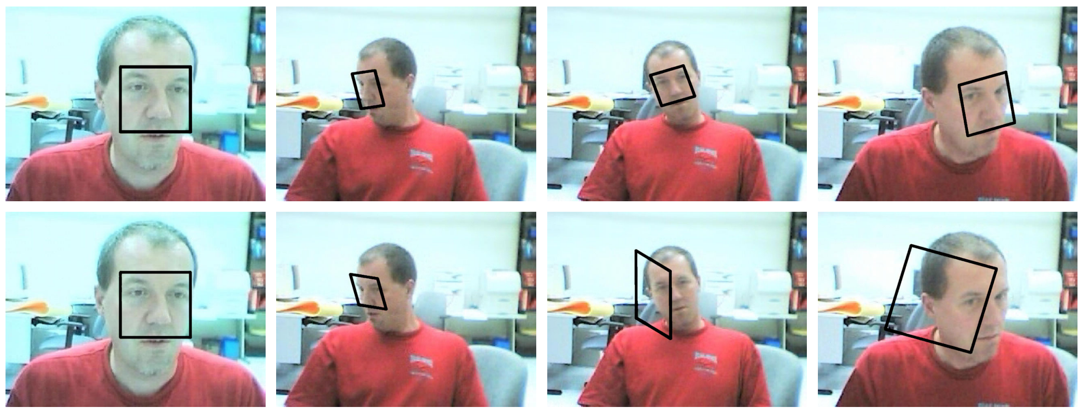


Figure 5.10: The effect of model update. The *first row* is with update. The *second row* is without update. The target has severe appearance variations due to illumination and pose change throughout the sequence. The sequence contains 324 frames.

method on the estimations. The sequence has large appearance variations, but the update method adapts to these changes (first row), whereas without update (second row) the target is lost.

In Figures 5.11 and 5.12 we show tracking examples for two challenging sequences. The targets have large on-plane and off-plane rotations, translations, scale changes, appearance changes and occlusions. The estimations are accurate, which shows the robustness of the proposed approach.

5.6.2 Invariant Object Detection

We perform detection experiments on a face dataset which consists of 803 face images from CMU, MIT and MERL datasets. The dataset is divided into 503 images for training and 300 for testing. The training set consists of 25150 samples which are generated by applying 50 transformations having a random norm between 0 to 1.0 to each face image. In contrast to the previous section, each of the six generators on the Lie algebra are weighted. For example, on average a norm 1.0 transformation corresponds to a combined translation of 25% of object size, rotation of $\pi/4$ degrees, and scaling and sheer ratio of 1.7.

In the *first experiment*, we compare the ridge and regression forest models either parameterized on the Lie algebra or based on the linearization. For each test image,

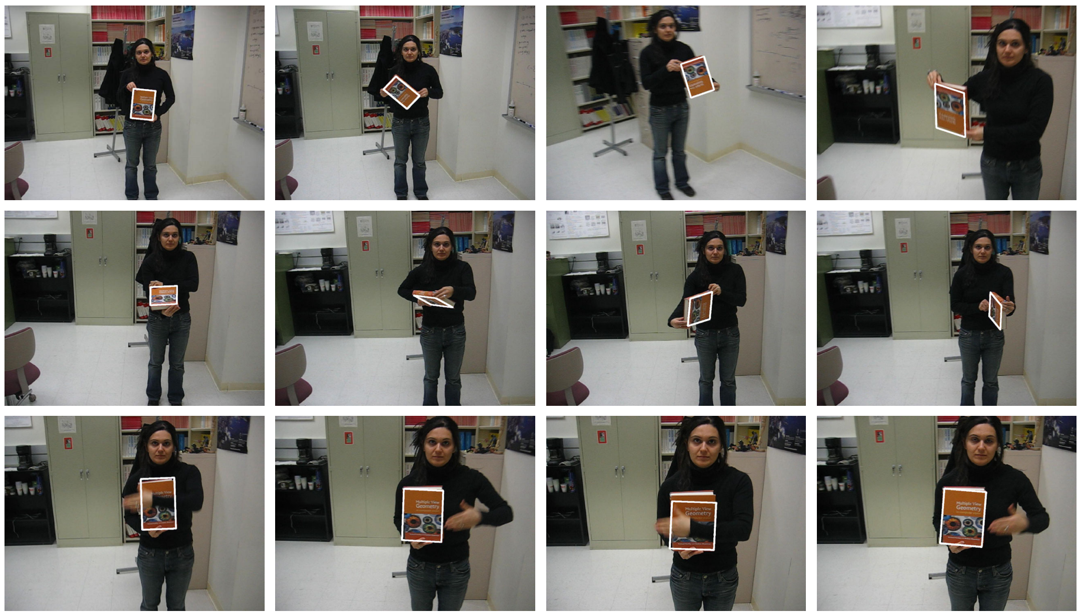


Figure 5.11: Affine tracking of a book. The target has large on-plane and off-plane rotations, translations, scale changes and occlusions. The sequence contains 739 frames.

we initialize five times the tracking window with a fixed norm random transformation (between 0.0 to 1.2) from the original location. For each initialization we perform 20 iterations of tracking. At the final location, we measure the squared geodesic error (5.18) from the original location. The mean squared geodesic errors are given in Figure 5.13. The Lie algebra based parametrization is significantly better for both regression models, especially for large transformations. The best result is given by the regression forest model.

In the *second experiment*, we compare the detection performances of the models. Using the setting of the first experiment, at the final locations we evaluate the face detector (Figure 5.14). The Viola and Jones (VJ) face detector [165] evaluated at the original location of the target could detect 96.7% of the faces, and the detection rate suddenly falls to 5% at locations which are norm 0.5 distant. The Lie algebra based estimations and the regression forest model are significantly superior. For norms between 0.0 to 1.0, the average miss rate for regression forest using Lie algebra is 4.24% which is almost as good as detector applied at the original location. For small transformations, the results are even better than the detector evaluated at the original

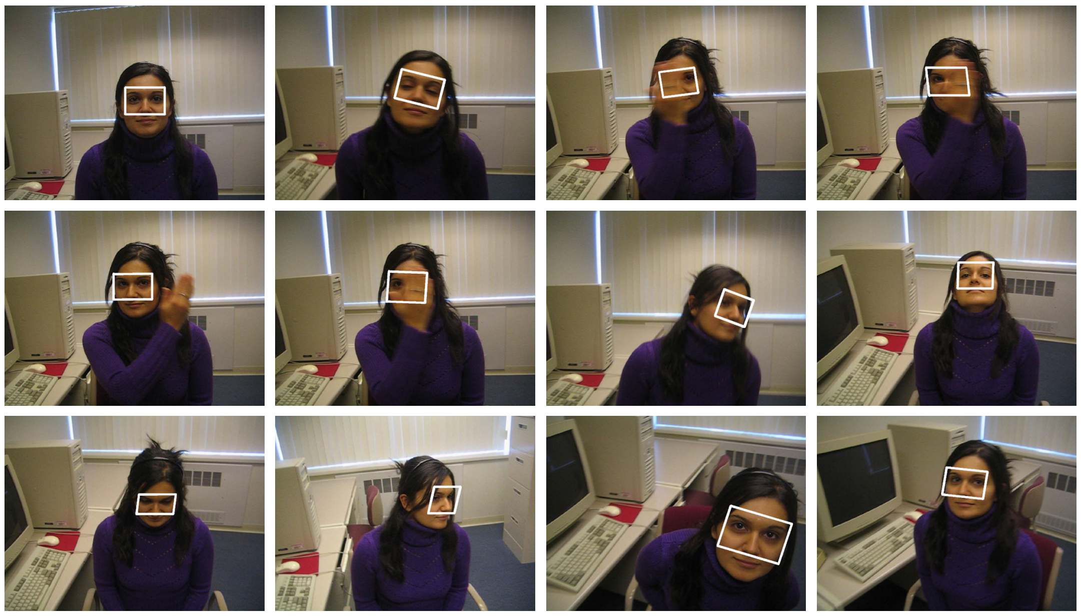


Figure 5.12: Affine tracking of a face. The target has large on-plane and off-plane rotations, translations, scale, illumination and appearance changes and occlusions. The sequence contains 425 frames.

location, where the small misalignments in the test set are corrected. On average, Lie algebra based parametrization have 50% less miss rate for regression forest and 24% less for ridge regression, compared to linearization. In Figure 5.15, several examples of initial and final locations found by the tracker are shown for various face images using regression forest model for transformations of norm 1.2.

In Figure 5.16 we show face detection examples for several challenging images utilizing both the original VJ detector and the proposed method. For an image of size 320×240, the VJ detector evaluates 58367 locations for translation and scale search, whereas the proposed method evaluates face detector at only 642 locations searched on the affine space.

5.6.3 Computational Requirement

The model is implemented on a Pentium D 2.80Ghz processor with 2.00GB of RAM using C++. The proposed tracking algorithm including model learning and update can process 60 frames per second.

The training of detection models requires 2 minutes for the ridge regression model

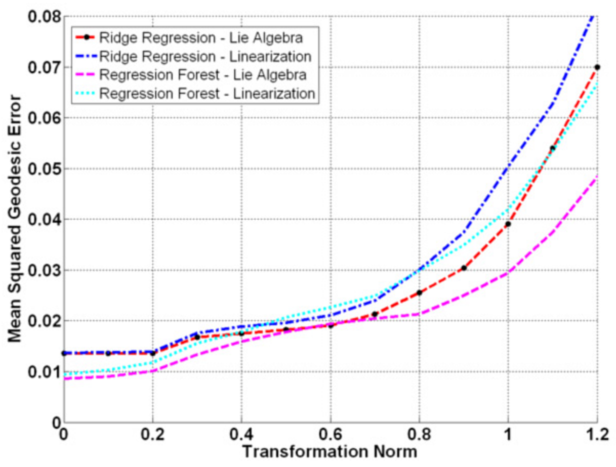


Figure 5.13: Estimation errors of ridge and regression forest models parameterized on the Lie algebra and the linearization.

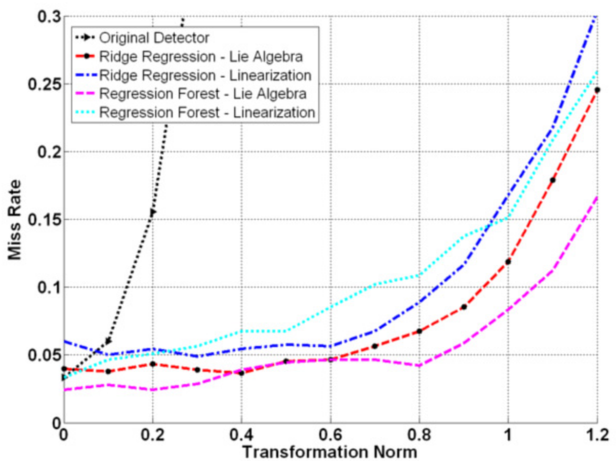


Figure 5.14: Miss rates of ridge and regression forest models parameterized on the Lie algebra and the linearization.

and around 3 hours for the random forest model. At runtime the most expensive operation is the affine warping of the regions to the object coordinates and computing the object descriptors. For a 320×240 image, the tracker is initialized at 642 locations and 20 tracker iterations are performed which requires 12840 warping operations. Since the warps at each iteration can be performed in parallel we implemented the warping in GPU using NVIDIA GeForce 8800 GTX graphics card and CUDA SDK. The search for an image of size 320×240 takes 0.85 and 2.4 seconds with the linear and the regression forest models respectively.

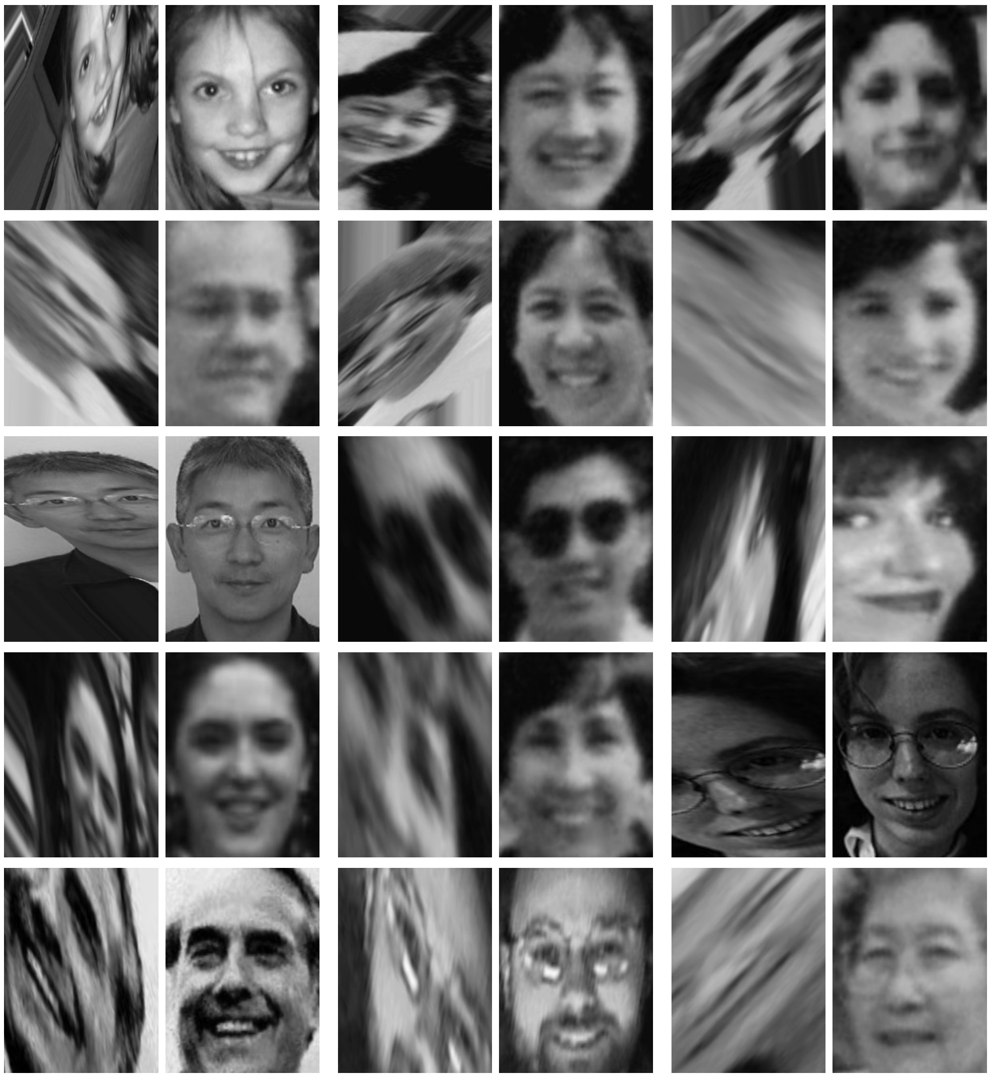


Figure 5.15: Face tracking. Columns 1, 3 and 5 are the initial windows and 2, 4 and 6 are the recovered locations.

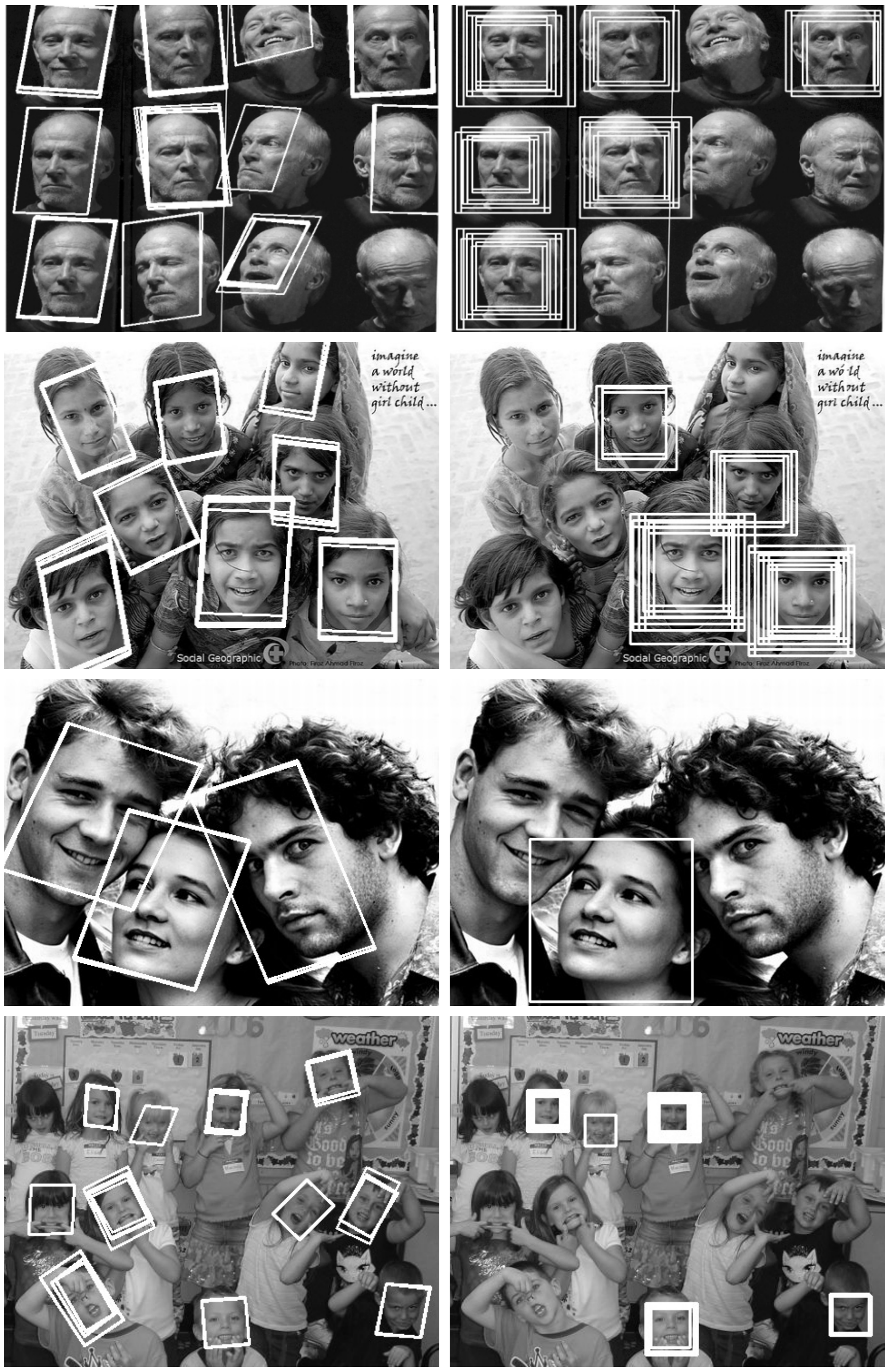


Figure 5.16: Face detection examples using the proposed method (*first column*) and the original VJ detector with a dense scan (*second column*). The original VJ detector is very sensitive to pose changes.

Chapter 6

Bayesian Background Modeling and Low Frame Rate Tracking

6.1 Introduction

Segmentation of foreground and background regions in video is a fundamental task in computer vision. The provided information is important for higher level operations such as object tracking.

The trivial approach to detect moving regions in image sequences is to select a reference frame while scene is stationary and to subtract the observed frames from this image. The resulting difference image is thresholded to extract the moving regions. Although this task looks fairly simple, in real world applications the approach rarely works. Usually the scene background is never static and varies by time due to several reasons. The most important factors are illumination changes, moving regions and the camera noise. Moreover in many applications, it is desirable to model the variable appearances of the background such as shadows.

To overcome these problems adaptive background models were proposed. The adaptive models can be categorized into two groups. The first group of methods use sequential filters to predict the background intensities. In [91, 92] Kalman filter is used to model background dynamics. Similarly in [151], Wiener filter is used to make a linear prediction of the intensity, given the pixel histories.

The second group of methods estimate the probability distributions of the pixel intensities. In [172], a parametric background model is presented by modeling the intensities of each pixel by a univariate normal distribution. The densities are updated online with the new measurements by a simple average filter.

In general the the scene background is not unimodal. Sudden illumination changes, moving and shadowed regions are some of the sources of multimodalities. The idea of using per pixel Gaussian distribution is extended by Gaussian mixture model (GMM) to represent the different appearances of the scene. In [55], the color distribution of each pixel is modeled with a mixture of three Gaussians corresponding to road, vehicle and shadow, for a traffic surveillance application. Likewise, Stauffer and Grimson [146] used mixture of k normal distributions and the model parameters are updated using an online expectation maximization (EM) algorithm. In [69], Harville et. al. extends the color model with the depth information coming from stereo cameras. In [81] and [83] gradient information is utilized to achieve a more accurate background subtraction.

Although any distribution can be represented with a GMM provided enough number of components, this is not computationally feasible for real time applications. In practice, it is only possible to represent each pixel with three to five components. In [44], a nonparametric background model is presented. The model keeps a history window of recent observations and the intensity distribution is approximated via kernel density estimation. Background subtraction is performed by thresholding the probability of the observed samples. In a more recent nonparametric approach [111], motion information is utilized to model dynamic scenes in addition to the intensity information. The drawback of the nonparametric approaches is the increased computational and memory requirements which is linear in the size of the temporal window. The other approaches include sequence classification using hidden Markov models (HMMs). In [130], three states corresponding to foreground, background and shadow is defined, whereas in [147] topology is learned from the observations.

Here we describe a new method for modeling background statistics of a dynamic scene [153]. Each pixel is represented with layers of Gaussian distributions. Using recursive Bayesian learning, we estimate the posterior distribution of mean and covariance of each layer. The proposed algorithm preserves the multimodality of the background and estimates the number of necessary layers for representing each pixel.

Tracking objects in low frame rate video is a challenging problem. In static camera configurations, background information is an important source to improve the tracking performance. Integrating the background information to mean shift tracker we present a robust low frame rate tracker [125].

6.2 Background Model

Our background model is most similar to adaptive mixture models [146], but instead of mixture of Gaussian distributions we define each pixel as layers of 3D multivariate Gaussians. Each layer corresponds to a different appearance of the pixel. We perform our operations on (r,g,b) color space.

Using Bayesian approach, we estimate the posterior distributions of mean and variance of each layer. We can extract necessary statistical information, regarding to these parameters from the density functions. For now, we are using expectations of mean and covariance for change detection and the covariance of the mean for confidence.

Prior knowledge can be integrated to the system easily with prior parameters. Due to computation of full covariance matrix, feature space can be modified to include other information sources, such as motion information, as discussed in [111].

Our update algorithm maintains the multimodailty of the background model. At each update, at most one layer is updated with the current observation. This assures the minimum overlap over layers. We also determine how many layers are necessary for each pixel and use only those layers during foreground segmentation phase. This is performed with an embedded confidence score. Details are explained in the following sections.

6.2.1 Layer Model

Data is assumed to be normally distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Mean and covariance are assumed unknown and modeled as random variables [60, p.87-88]. Using Bayes theorem joint posterior density can be written as

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{x}) \propto p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})p(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{6.1}$$

To perform recursive Bayesian estimation with the new observations, joint prior density $p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ should have the same form with the joint posterior density $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{x})$. Conditioning on the variance, joint prior density is written as

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\boldsymbol{\mu}|\boldsymbol{\Sigma})p(\boldsymbol{\Sigma}). \tag{6.2}$$

Above condition is realized if we assume inverse Wishart distribution for the covariance, and conditioned on the covariance, multivariate normal distribution for the mean. Inverse Wishart distribution is a multivariate generalization of scaled inverse-χ^2 distribution. The parametrization is given by

$$\boldsymbol{\Sigma} \sim \text{Inv-Wishart}_{\upsilon_{t-1}}(\boldsymbol{\Lambda}_{t-1}^{-1}) \tag{6.3}$$

$$\boldsymbol{\mu}|\boldsymbol{\Sigma} \sim \text{N}(\boldsymbol{\theta}_{t-1}, \boldsymbol{\Sigma}/\kappa_{t-1}). \tag{6.4}$$

where υ_{t-1} and $\boldsymbol{\Lambda}_{t-1}$ are the degrees of freedom and scale matrix for inverse Wishart distribution, $\boldsymbol{\theta}_{t-1}$ is the prior mean, κ_{t-1} is the number of prior measurements. With these assumptions joint prior density becomes

$$\begin{aligned} p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \quad & |\boldsymbol{\Sigma}|^{-((\upsilon_{t-1}+3)/2+1)} \times \\ & e^{\left(-\frac{1}{2}tr(\boldsymbol{\Lambda}_{t-1}\boldsymbol{\Sigma}^{-1}) - \frac{\kappa_{t-1}}{2}(\boldsymbol{\mu}-\boldsymbol{\theta}_{t-1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\theta}_{t-1})\right)} \end{aligned} \tag{6.5}$$

for three dimensional space. Let this density be labeled as normal-inverse-Wishart with parameters $(\boldsymbol{\theta}_{t-1}, \boldsymbol{\Lambda}_{t-1}/\kappa_{t-1}; \upsilon_{t-1}, \boldsymbol{\Lambda}_{t-1})$. Multiplying prior density with the normal likelihood and arranging the terms, joint posterior density becomes normal-inverse-Wishart$(\boldsymbol{\theta}_t, \boldsymbol{\Lambda}_t/\kappa_t; \upsilon_t, \boldsymbol{\Lambda}_t)$ with the parameters updated as

$$\upsilon_t = \upsilon_{t-1} + n \qquad \kappa_n = \kappa_{t-1} + n \tag{6.6}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}\frac{\kappa_{t-1}}{\kappa_{t-1}+n} + \overline{\mathbf{x}}\frac{n}{\kappa_{t-1}+n} \tag{6.7}$$

$$\begin{aligned} \boldsymbol{\Lambda}_t = \; & \boldsymbol{\Lambda}_{t-1} + \sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T + \\ & n\frac{\kappa_{t-1}}{\kappa_t}(\overline{\mathbf{x}} - \boldsymbol{\theta}_{t-1})(\overline{\mathbf{x}} - \boldsymbol{\theta}_{t-1})^T \end{aligned} \tag{6.8}$$

where $\overline{\mathbf{x}}$ is the mean of new samples and n is the number of samples used to update the model. If update is performed at each time frame, n becomes one. To speed up the system, update can be performed at regular time intervals by storing the observed samples. During our tests, we update one quarter of the background at each time frame, therefore n becomes four. The new parameters combine the prior information with the observed samples. Posterior mean $\boldsymbol{\theta}_t$ is a weighted average of the prior mean and the sample mean. The posterior degrees of freedom is equal to prior degrees of freedom plus the sample size. System is started with the following initial parameters

$$\kappa_0 = 10, \quad \upsilon_0 = 10, \quad \boldsymbol{\theta}_0 = \mathbf{x}_0, \quad \boldsymbol{\Lambda}_0 = (\upsilon_0 - 4)16^2\mathbf{I} \tag{6.9}$$

where $\mathbf{I}$ is the three dimensional identity matrix.

Integrating joint posterior density with respect to $\boldsymbol{\Sigma}$ we get the marginal posterior density for the mean

$$p(\boldsymbol{\mu}|\mathbf{x}) \propto t_{\upsilon_t-2}(\boldsymbol{\mu}|\boldsymbol{\theta}_t, \boldsymbol{\Lambda}_t/(\kappa_t(\upsilon_t - 2))) \tag{6.10}$$

where t_{υ_t-2} is a multivariate t-distribution with $\upsilon_t - 2$ degrees of freedom.

We use the expectations of marginal posterior distributions for mean and covariance as our model parameters at time t. Expectation for marginal posterior mean (expectation of multivariate t-distribution) is given by

$$\boldsymbol{\mu}_t = E(\boldsymbol{\mu}|\mathbf{x}) = \boldsymbol{\theta}_t \tag{6.11}$$

whereas expectation of marginal posterior covariance (expectation of inverse Wishart distribution) becomes

$$\boldsymbol{\Sigma}_t = E(\boldsymbol{\Sigma}|\mathbf{x}) = (\upsilon_t - 4)^{-1}\boldsymbol{\Lambda}_t. \tag{6.12}$$

Our confidence measure for the layer is equal to one over determinant of covariance of $\boldsymbol{\mu}|\mathbf{x}$

$$C = \frac{1}{|\boldsymbol{\Sigma}_{\boldsymbol{\mu}|\mathbf{x}}|} = \frac{\kappa_t^3(\upsilon_t - 2)^4}{(\upsilon_t - 4)|\boldsymbol{\Lambda}_t|}. \tag{6.13}$$

If our marginal posterior mean has larger variance, our model becomes less confident. Note that variance of multivariate t-distribution with scale matrix $\boldsymbol{\Sigma}$ and degrees of freedom υ is equal to $\frac{\upsilon}{\upsilon-2}\boldsymbol{\Sigma}$ for $\upsilon > 2$.

System can be further speed up by making independence assumption on color channels. Update of full covariance matrix requires computation of nine parameters. Moreover, during distance computation we need to invert the full covariance matrix. To speed up the system, we separate (r, g, b) color channels. Instead of multivariate Gaussian for a single layer, we use three univariate Gaussians corresponding to each color channel. After updating each color channel independently we join the variances and create a diagonal covariance matrix:

$$\mathbf{\Sigma}_t = \begin{pmatrix} \sigma^2_{t,r} & 0 & 0 \\ 0 & \sigma^2_{t,g} & 0 \\ 0 & 0 & \sigma^2_{t,b} \end{pmatrix}. \tag{6.14}$$

In this case, for each univariate Gaussian we assume scaled inverse-χ^2 distribution for the variance and conditioned on the variance univariate normal distribution for the mean. The Bayesian update equations for the parameters can be found in [60, p.78-80].

6.2.2 Background Update

We initialize our system with k layers for each pixel. Usually we select three to five layers. In more dynamic scenes more layers are required. As we observe new samples for each pixel, we update the parameters for our background model. We start our update mechanism from the most confident layer in our model. If the observed sample is inside the 99% confidence interval of the current model, parameters of the model are updated as explained in equations (6.6), (6.7) and (6.8). Lower confidence models are not updated.

For background modeling, it is useful to have a forgetting mechanism so that the earlier observations have less effect on the model. Forgetting is performed by reducing the number of prior observations parameter of unmatched model. If current sample is not inside the confidence interval we update the number of prior measurements parameter via

$$\kappa_t = \kappa_{t-1} - n \tag{6.15}$$

and proceed to the update of next confident layer. We do not let κ_t become less than the initial value 10. If none of the models are updated, we delete the least confident layer and initialize a new model having current sample as the mean and an initial variance (6.9). The update algorithm for a single pixel is given in Figure 6.1.

```
Input: New sample $\mathbf{x}$, current background layers $\{(\boldsymbol{\theta}_{t-1,i}, \boldsymbol{\Lambda}_{t-1,i}, \kappa_{t-1,i}, \upsilon_{t-1,i})\}_{i=1..k}$

• Sort layers according to confidence measure defined in (6.13)
• i=1
• While $i < k$
    – Measure Mahalanobis distance from layer $i$
      $(\mathbf{x} - \boldsymbol{\mu}_{t-1,i})^T \boldsymbol{\Sigma}_{t-1,i}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{t-1,i})$
    – If sample $\mathbf{x}$ is in 99% confidence interval
        ∗ update model parameters according to equations (6.6), (6.7), (6.8) and STOP
    – Else
        ∗ Update model parameters according to equation (6.15)
    – i=1+1
• Delete layer $k$, initialize a new layer having parameters defined in equation (6.9)
```

Figure 6.1: Background update algorithm.

With this mechanism, we do not deform our models with noise or foreground pixels, but easily adapt to smooth intensity changes like illumination. Embedded confidence score determines the number of layers to be used and prevents unnecessary layers. During our tests usually secondary layers corresponds to shadowed form of the background pixel or different colors of the moving regions of the scene. If the scene is unimodal, confidence scores of layers other than first layer becomes very low.

6.2.3 Comparison with Online EM

In [146], each pixel is represented with a mixture of Gaussian distribution. The parameters of the Gaussians and the mixing coefficients are updated with an online k-means approximation to expectation maximization (EM). The approach is very sensitive to initial observations. If the components are improperly initialized, every component eventually converges to the most significant mode of the distribution. In general the smaller modes nearby larger modes are never detected.

To demonstrate the performance of the algorithm, we generated 1D data from mixture of Gaussian distribution corrupted with uniform noise. First data set consists of

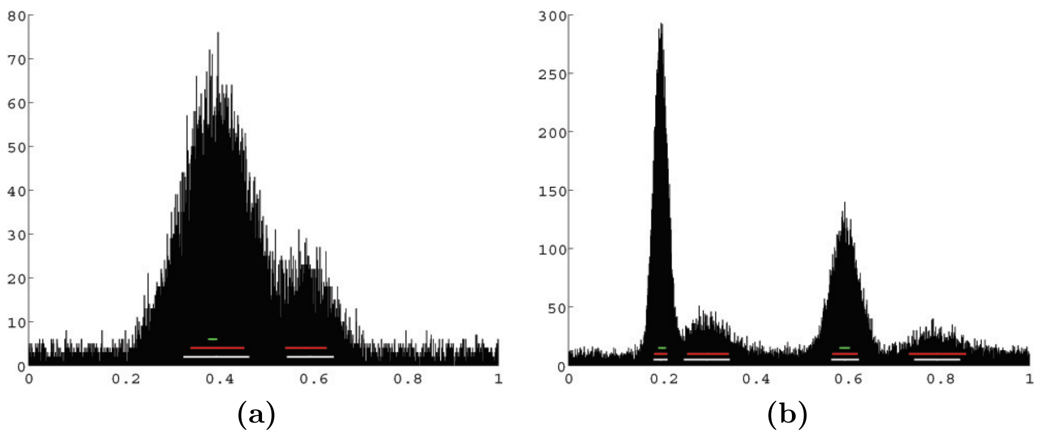


Figure 6.2: 1D Mixture of Gaussian data corrupted with uniform noise. Lines show one standard deviation interval around the mean. Parameters are estimated with recursive Bayesian learning and online EM [146] with five Gaussians. White line is the real parameters. Red line shows estimation with recursive Bayesian learning. Green line shows estimation with online EM. (a) Mixture of two Gaussians. Most confident two layers estimated by two methods are shown. (b) Mixture of four Gaussians. Most confident four layers estimated by two methods are shown. There are multiple Gaussians at the same place in online EM and some modes are not detected.

12000 points corrupted with 3000 noise samples and second data set consists of 23000 points corrupted with 10000 noise samples. We assume that we observe the data in random order. We treat the samples as observations coming from a single pixel and estimate the model parameters with our approach and online EM algorithm. One standard deviation interval around the mean for actual and estimated parameters are plot on the histogram, in Figure 6.2. Results show that, with online EM estimation the multimodality is lost and the models converge to the most significant modes. With our method, multimodality of the distribution is maintained. Another important observation is, estimated variance with online EM algorithm is always much smaller than the actual variance. This is not surprising since the update is proportional to the likelihood of the sample, so samples closer to the mean become more important.

Numerical estimations for mean and variance, and the normalized confidence scores are shown in Tables 6.1 and 6.2. Our confidence score is very effective in determining the number of necessary layers for each pixel. Although we estimate the model parameters with five layers, it is clear from our confidence scores that how many layers are effective. There is a large gap between significant and insignificant layers.

		Mode1	Mode2	Mode3	Mode4	Mode5
Real	Num.	10000	2000			
	Mean	0.4000	0.6000			
	Std.	0.0700	0.0500			
EM	Mean	0.3923	0.3919	0.3919	0.3919	0.4545
	Std.	0.0093	0.0093	0.0093	0.0093	0.0631
	Conf.	0.2538	0.2482	0.2481	0.2481	0.0016
Bayes	Mean	0.4021	0.5906	0.8488	0.2561	0.1133
	Std.	0.0572	0.0440	0.0820	0.0268	0.0670
	Conf.	0.7047	0.2519	0.0214	0.0208	0.0009

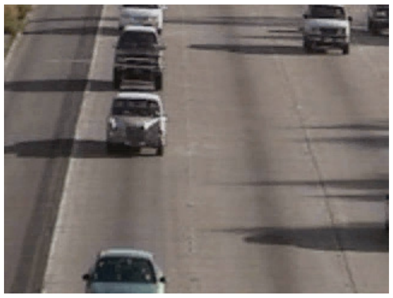
Table 6.1: Mixture of two Gaussians.

		Mode1	Mode2	Mode3	Mode4	Mode5
Real	Num.	10000	8000	3000	2000	
	Mean	0.2000	0.6000	0.3000	0.8000	
	Std.	0.0150	0.0300	0.0500	0.0500	
EM	Mean	0.2033	0.2033	0.5993	0.5993	0.9382
	Std.	0.0085	0.0085	0.0113	0.0113	0.0633
	Conf.	0.3772	0.3772	0.1221	0.1221	0.0111
Bayes	Mean	0.2002	0.5998	0.3026	0.8004	0.9387
	Std.	0.0146	0.0277	0.0451	0.0620	0.0632
	Conf.	0.3996	0.3820	0.1088	0.1087	0.0007

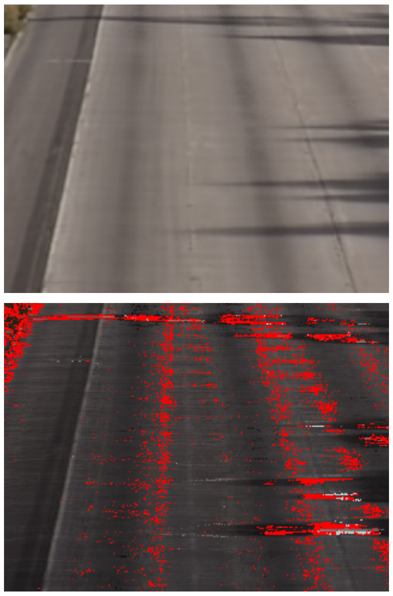
Table 6.2: Mixture of four Gaussians.

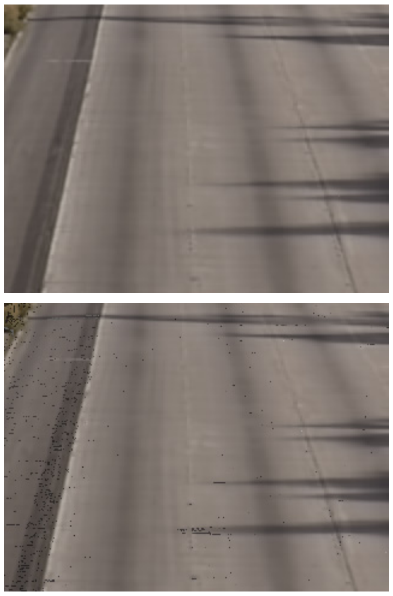
Real data results are presented in Figures 6.3 and 6.4 where the first sequence is a traffic sequence with heavy shadows and the second sequence is a dynamic outdoor scene. In the first sequence, first and second layers of our background corresponds to the original and shadowed versions of the background. The locations where most of the cars move have higher variances, so usually they are less confident. Those pixels are shown in red. In online EM algorithm, the first and the second layers converged to the most significant mode.

In the second sequence, most significant three layers estimated by two algorithms are shown. As seen in original images, the sky and the trees are changing appearance by time. Our background model successfully modeled the different appearances of these regions. The appearance of grass does not change much with time. As expected, confidence score of second and third layers of our background are very low for this region.



(a)





(b) **(c)**

Figure 6.3: Traffic video with heavy shadows. (a) A sample from the original sequence. (b) Most confident two layers with recursive Bayesian learning. (c) Most confident two layers with online EM. With recursive Bayesian learning, we are able to model the shadows as the second layer of the scene, whereas in EM first and second layers converge to the most significant mode.

6.2.4 Foreground Segmentation

Learned background statistics are used to detect the changed regions of the scene. Number of layers required to represent a pixel is not known beforehand so background is initialized with more layers than needed. Using the confidence scores we determine how many layers are significant for each pixel. We order the layers according to confidence score (6.13) and select the layers having confidence value greater than the layer threshold T_c. We refer to these layers as confident layers. In our experiments we use $T_c = 1.0$.

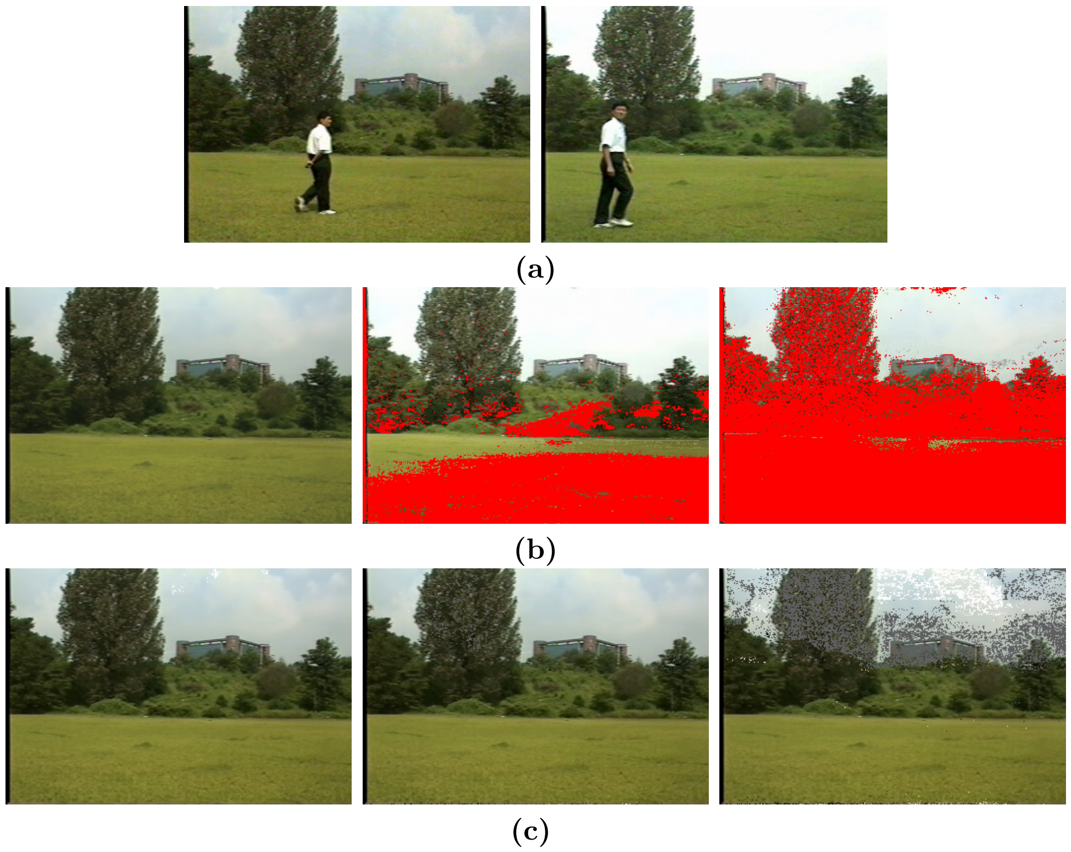
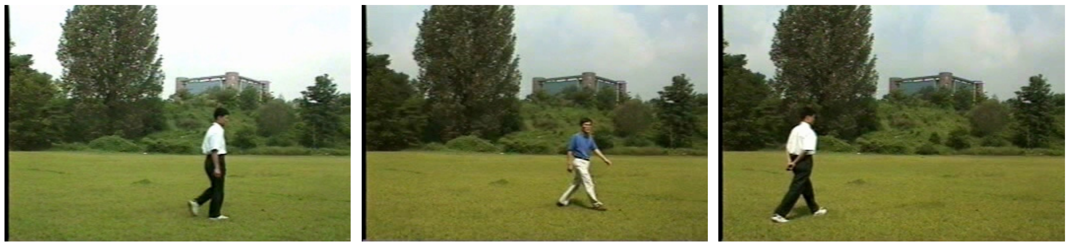


Figure 6.4: Outdoor video. (a) Samples from the original sequence. (b) First three layers of recursive Bayesian learning. Different appearances of the background is captured with first three layers. Red pixels are nonconfident layers. (c) First three layers of online EM. Second and third layers are almost same with first layer.

We measure the Mahalanobis distance of observed color from the confident layers. Pixels that are outside of 99% confidence interval of all confident layers of the background are considered as foreground pixels.

In Figure 6.5, we present foreground segmentation results of a dynamic scene. As seen in Figure 6.5a, appearance of background is changing with time. After several frames, our background algorithm learns the different appearances of the background. Although the method is very sensitive to foreground objects, it does not produce false alarms in sudden background changes (Figure 6.5b).


(a)

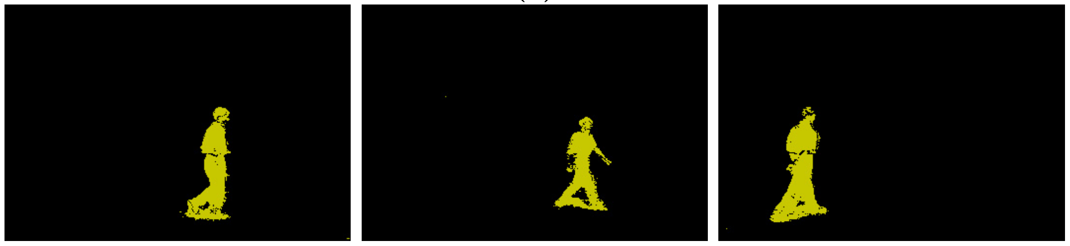

(b)

Figure 6.5: Foreground segmentation of a dynamic scene. (a) Original sequence. (b) Detected foreground pixels.

6.3 Low Frame Rate Tracking

We use a modified version of the mean shift tracker [29] to estimate the location of the target. The original mean shift tracker requires significant overlap on the target kernels in consequent frames. In low frame rate data, target movements are usually large and unpredictable, so single mean shift window centered at the previous location of the target is not enough. To overcome this problem, we initialize the mean shift tracker both on the previous location of the target and the high motion areas of the scene detected through foreground segmentation. Final location of the target is determined by the maximum of the object likelihood scores computed at the converged locations.

6.3.1 Object Model

Object model is a nonparametric color template. The template is a $(W \times H) \times D$ tensor whose elements are 3D color samples from the object, where W and H are the width and height of the template respectively and D is the size of the history window. We refer to the pixels inside the estimated target box as $\{(\mathbf{x}_i, \mathbf{u}_i)\}_{i=1\ldots N}$, where $\mathbf{x}_i$ is the 2D coordinate in the image coordinate system and $\mathbf{u}_i$ is the 3D color vector. Corresponding sample points in the template are referred as $\{(\mathbf{y}_j, \mathbf{v}_{jk})\}_{j=1\ldots M}$, where

$\mathbf{y}_j$ is the 2D coordinates in the template coordinate system and $\{\mathbf{v}_{jk}\}_{k=1..D}$ are the 3D color values. Recall that index i inside the estimated target box maps to index j in the template. This mapping is not one-to-one. Usually, size of the target box is much larger than the size of the template, so one pixel in the template maps to several pixels inside the target box. During tracking, we replace the oldest sample of each pixel of the template with one corresponding pixel from the image. Using foreground segmentation (Section 6.2.4), template pixels which correspond to the background pixels in the current frame are not updated.

6.3.2 Mean shift with Background Information

Although color histogram based mean shift algorithm is efficient and robust for nonrigid object tracking, if the tracked object's color is similar to the background, tracking performance reduces. We integrate the background information to improve the tracking performance.

Let $\{q_s\}_{s=1..m}$ be the kernel weighted color histogram of the reference model. Reference model histogram is constructed using the nonparametric object template (Section 6.3.1)

$$q_s = Q_1 \sum_{j=1}^{M} \sum_{k=1}^{D} k_N \left(\left\| \frac{\mathbf{y}_j}{h_t} \right\|^2 \right) \delta(\hat{m}(\mathbf{v}_{jk}) - s) \tag{6.16}$$

where template bandwidth h_t is equal to half size of the template size (both horizontal an vertical) and k_N is the profile of the univariate normal kernel

$$k_N(x) = e^{-\frac{1}{2}x} \tag{6.17}$$

Constant term Q_1 satisfies that $\sum_{s=1}^{m} q_s = 1$ and the function $\hat{m}$ maps a color value to the corresponding histogram bin in the quantized color space. Object template has a history information which makes the histogram more accurate in occlusions. The kernel function assigns smaller weights to the pixels farther away from the center which are usually contaminated by noise. Let $\mathbf{p}(\mathbf{z})$ and $\mathbf{b}(\mathbf{z})$ be the kernel weighted color histogram of the image and the background centered at location $\mathbf{z}$. We construct background color histogram using only the confident layers of the background as explained in Section 6.2.4.

Bhattacharya coefficient [26], $\rho(\mathbf{p}(\mathbf{z}), \mathbf{q}) = \sum_{s=1}^{m} \sqrt{q_s p_s(\mathbf{z})}$, measures the similarity between the target histogram and histogram of the proposed location $\mathbf{z}$ in the current frame. We integrate the background information and define the new similarity function as

$$\psi(\mathbf{z}) = \alpha_f \rho(\mathbf{p}(\mathbf{z}), \mathbf{q}) - \alpha_b \rho(\mathbf{p}(\mathbf{z}), \mathbf{b}(\mathbf{z})) \tag{6.18}$$

where α_f and α_b are the mixing coefficients for foreground and background. Besides maximizing the target similarity, we penalize the similarity between the target and the background image histograms. The location where the target is, should have a different appearance than the background. We use $\alpha_f = 1$ and $\alpha_b = 1/2$. The similarity function can be rewritten as

$$\psi(\mathbf{z}) = \sum_{s=1}^{m} \sqrt{p_s(\mathbf{z})} \left(\alpha_f \sqrt{q_s} - \alpha_b \sqrt{b_s(\mathbf{z})} \right) \tag{6.19}$$

Let $\mathbf{z}_0$ be the initial location where we start search for the target location. Using Taylor expansion around the values of $p_s(\mathbf{z}_0)$ and $b_s(\mathbf{z}_0)$

$$\begin{aligned} \psi(\mathbf{z}) \approx & \sum_{s=1}^{m} \sqrt{p_s(\mathbf{z}_0)} \left(\alpha_f \sqrt{q_s} - \alpha_b \sqrt{b_s(\mathbf{z}_0)} \right) + \\ & \sum_{s=1}^{m} \frac{\alpha_f \sqrt{q_s} - \alpha_b \sqrt{b_s(\mathbf{z}_0)}}{2\sqrt{p_s(\mathbf{z}_0)}} (p(\mathbf{z}) - p(\mathbf{z}_0)) - \\ & \sum_{s=1}^{m} \frac{\alpha_b \sqrt{p_s(\mathbf{z}_0)}}{2\sqrt{b_s(\mathbf{z}_0)}} (b(\mathbf{z}) - b(\mathbf{z}_0)). \end{aligned} \tag{6.20}$$

Putting constant terms inside Q_2 we obtain

$$\begin{aligned} \psi(\mathbf{z}) \approx & \; Q_2 + \sum_{s=1}^{m} \frac{\alpha_f \sqrt{q_s} - \alpha_b \sqrt{b_s(\mathbf{z}_0)}}{2\sqrt{p_s(\mathbf{z}_0)}} p(\mathbf{z}) - \\ & \sum_{s=1}^{m} \frac{\alpha_b \sqrt{p_s(\mathbf{z}_0)}}{2\sqrt{b_s(\mathbf{z}_0)}} b(\mathbf{z}). \end{aligned} \tag{6.21}$$

Using the definition of $\mathbf{p}(\mathbf{z})$ and $\mathbf{b}(\mathbf{z})$, the similarity function is rewritten as

$$\psi(\mathbf{z}) \approx Q_2 + Q_3 \sum_{i=1}^{N} w_i k_N \left(\left\| \frac{\mathbf{z} - \mathbf{x}_i}{h} \right\|^2 \right), \tag{6.22}$$

$$\begin{aligned} w_i = & \sum_{s=1}^{m} \frac{\alpha_f \sqrt{q_s} - \alpha_b \sqrt{b_s(\mathbf{z}_0)}}{2\sqrt{p_s(\mathbf{z}_0)}} \delta[\hat{m}_f(\mathbf{x}_i) - s] - \\ & \sum_{s=1}^{m} \frac{\alpha_b \sqrt{p_s(\mathbf{z}_0)}}{2\sqrt{b_s(\mathbf{z}_0)}} \delta[\hat{m}_b(\mathbf{x}_i) - s]. \end{aligned} \tag{6.23}$$

where $\hat{m}_f()$ and $\hat{m}_b()$ maps a pixel in observed and background images, to the corresponding color bin in quantized color space. The spatial bandwidth h is equal to the half size of the candidate box along each dimension. The second term in (6.22) is equal to the kernel density estimation with data weighted by w_i. Mode of this distribution (maximum of similarity function (6.18)) can be found by mean shift algorithm. Recall that the weights w_i might be negative. Unlike [25] and [178], we use zero instead of the negative weights. Mean shift vector at location $\mathbf{z}_0$ becomes

$$\mathbf{m}(\mathbf{z}_0) = \frac{\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{z}_0) w_i g_N(\|\frac{\mathbf{z}_0 - \mathbf{x}_i}{h}\|^2)}{\sum_{i=1}^{n} w_i g_N(\|\frac{\mathbf{z}_0 - \mathbf{x}_i}{h}\|^2)}. \tag{6.24}$$

where $g_N(x) = -k'_N(x)$.

6.3.3 Template Likelihood

The probability of a pixel $(\mathbf{x}_i, \mathbf{u}_i)$ inside the candidate target box centered at $\mathbf{z}$ belongs to the object can be estimated with the Parzen window estimator

$$l_j(\mathbf{u}_i) = \frac{1}{Dh_c^3} \sum_{k=1}^{D} k_N \left(\left\| \frac{\mathbf{u}_i - \mathbf{v}_{jk}}{h_c} \right\|^2 \right). \tag{6.25}$$

Bandwidth of the color kernel is selected as $h_c = 16$ for each color channel. The likelihood of an object being at location $\mathbf{z}$ is then measured by

$$L(\mathbf{z}) = \frac{1}{N} \sum_{i=1}^{N} l_j(\mathbf{u}_i) k_N \left(\left\| \frac{\mathbf{x}_i - \mathbf{z}}{h} \right\|^2 \right). \tag{6.26}$$

The spatial kernel assigns smaller weights to samples farther from the center making the estimation more robust.

6.3.4 Scale Adaptation

Scale adaptation of the objects are performed using the foreground pixels. Let B be the box of the object centered at estimated location $\mathbf{z}$. We define a second box $\hat{B}$ around the object center which has twice area of B. We are trying to maximize

$$\sum_{\mathbf{x} \in B} fg(\mathbf{x}) + \sum_{\mathbf{x} \in \hat{B}/B} (1 - fg(\mathbf{x})) \tag{6.27}$$

```
Input: Target at location $\mathbf{z}_{t-1}$ at previous frame and the high motion
areas detected through foreground segmentation $\{\mathbf{c}_i\}_{i=1..L}$
  • $L_{max} = 0$
  • For $\mathbf{z} = \mathbf{z}_{t-1}$ and $\mathbf{z} = \mathbf{c}_i, \quad i = 1...L$
      – Compute mean shift vector $\mathbf{m}(\mathbf{z})$ using (6.24)
      – While $\psi(\mathbf{z}) < \psi(\mathbf{z} + \mathbf{m}(\mathbf{z}))$
          ∗ $\mathbf{z} = \mathbf{z} + \mathbf{m}(\mathbf{z})$
          ∗ Compute mean shift vector $\mathbf{m}(\mathbf{z})$ using (6.24)
      – Compute likelihood $L(\mathbf{z})$ using (6.26)
      – If $L(\mathbf{z}) > L_{max}$
          ∗ $\mathbf{z}_t = \mathbf{z}$, $L_{max} = L(\mathbf{z})$
  • Update scale via maximum of (6.27)
```

Figure 6.6: Low frame rate tracking algorithm.

where $fg(\mathbf{x})$ is one if $\mathbf{x}$ is a foreground pixel and zero otherwise. At each frame, leaving $\hat{B}$ fixed we modify B $\pm 5\%$ in all dimensions and chose the scale which gives the best score.

6.3.5 Tracking Algorithm

The mean shift tracker is initialized both on the previous location of the target $\mathbf{z}_{t-1}$ and the high motion areas of the scene $\{\mathbf{c}_i\}_{i=1..L}$, detected through foreground segmentation. The algorithm iterates by moving along the mean shift vector (6.24) which also includes the background information. When the iterations end we evaluate the likelihood score (6.26) at the converged location. The target location is given by the maximum of the likelihood function. Finally we update the size of the object via maximum of (6.27). The tracking algorithm is given in Figure 6.6.

Figure 6.7 shows a low frame rate tracking example (2 frames per second). Usually, there is no overlap on target kernels in two consecutive frames, therefore the original mean shift algorithm loses the targets after a few frames. The results show that, our template likelihood score is effective in resolving ambiguities caused by the multiple objects in the scene. The appearance of the targets are similar to the background. Fusion of background information improves the histogram based mean shift tracker significantly, which results in good target localization.

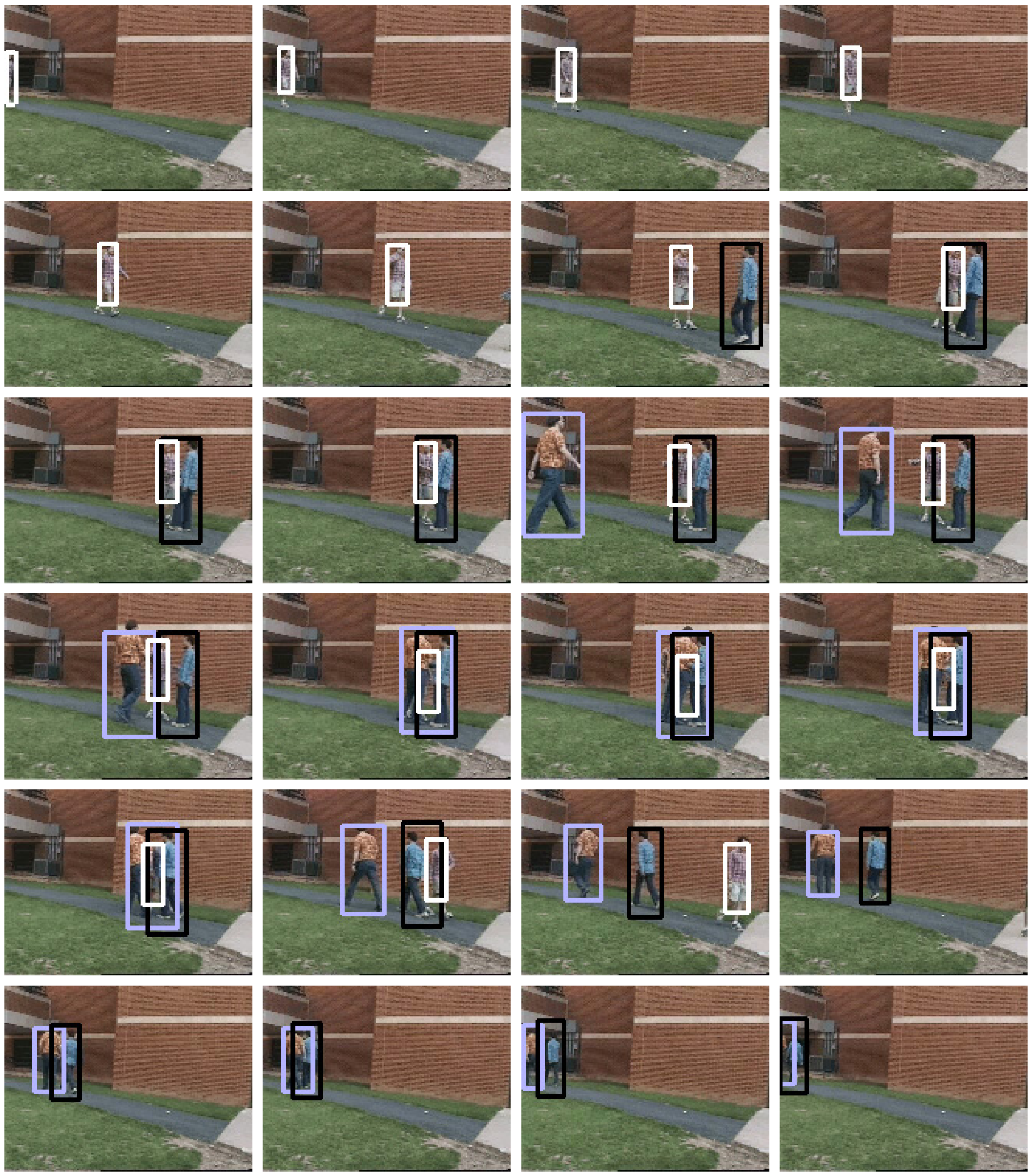


Figure 6.7: Low frame rate tracking example. The sampling rate is 2 frames per second.

Chapter 7

Classification of Hematologic Malignancies for Clinical Diagnosis

7.1 Introduction

Over the past few years there has been increased interest and efforts applied to utilizing content-based image retrieval in medical applications [70, 78, 82, 94, 95, 140]. Individual strategies and approaches differ according to the degree of generality (general purpose versus domain specific), the level of feature abstraction (primitive features versus logical features), overall dissimilarity measure used in retrieval ranking, database indexing procedure, level of user intervention (with or without relevance feedback), and by the methods used to evaluate their performance. Here we present a decision system utilizing texture based representation and support vector machine (SVM) optimization to classify hematologic malignancies [159].

The use of texture analysis for performing automated classification of disease based on features extracted from radiological imaging studies has been reported in the medical literature repeatedly. It has been successfully applied in breast cancer [106, 114, 142], liver cancer [77] and obstructive lung disease [19].

Recently there have been a number of investigators who have begun to explore the feasibility of utilizing texture features in the classification of pathology at the microscopic level. The success of the methods vary with the domain of the problem, and the choice of representation and optimization techniques. We also note that the testing methodology that is applied during the experiments and the size of the datasets have very important role in the observed results, and some test methodologies and small datasets can trigger biased results.

In [129], frequency domain features are used to classify among subclasses of normal and abnormal cervical cell images using a database containing 110 cells from both normal and abnormal groups. Using the spectra of cell images, 27 texture features are extracted with gray level difference method and together with 22 frequency components, resulted in 92% correct classification among subclasses. In [167], statistical geometric features, which are computed from several binary thresholded versions of texture images are used to classify among normal and abnormal cervical cells. The method gave 93% correct classification rate on a database containing 117 cervical cells using only 9 statistical features.

A diagnosis scheme among the main subsets of lung carcinomas was reported in [139] by chromatin texture feature analysis using a test set comprised of 195 specimens. Texture features describing the granularity and the compactness of the nuclear chromatin were extracted for calculation of classification rules, which allowed the discrimination of different tumor groups. Although the classification failed to distinguish among some subtypes of tumor groups, around 90% classification accuracy was achieved for small-cell and non-small-cell lung carcinoma using the four dimensional texture features combined with simple decision rules.

In [116], adaptive texture features were described utilizing the class distances and differences of gray level cooccurrence matrices of texture images. In [117], the adaptive texture features were used for classifying the nuclei of cells in ovarian cancer. In this study a clear relation between nuclear DNA content, area, first-order statistics, and texture is observed. The approach discriminated the two classes of cancer with a correct classification rate of 70% on a test set of 134 cases.

A decision support system to discriminate among three types of lymphoproliferatie disorders and normal blood cells was presented in [28]. Cells were represented with their nuclear shape, texture and area, where shape was characterized through similarity invariant Fourier descriptors and multiresolution simultaneous autoregressive model was utilized for texture description. The experiments conducted on a database consisting of 261 specimens using ten-fold cross validation were resulted with 80% correct classification rate. Notice that in all the methods described, the results were acquired using small datasets and without obeying the separation of cells among training-testing sets based on patient. We present a detailed discussion about the testing methodologies and their effects on the results in Section 7.5.

As new treatments emerge, each targeting specific clinical profiles, it becomes increasingly important to distinguish among subclasses of pathologies. In modern diagnostic pathology, sophisticated analyses are often needed to support a differential diagnosis, but these supporting tests are not typically employed unless morphological assessment of a specimen first leads one to classify the case as suspicious. In many cases, the differential diagnosis can only be rendered after immunophenotyping and/or molecular or cytogenetic study of the cells involved. Immunophenotyping is the process commonly used to analyze and sort lymphocytes into subsets based on their antigens using flow cytometry. For the purposes of our experiments the immunophenotype provide independent confirmation of the diagnosis for all cases. The additional studies are expensive, time consuming, and usually require fresh tissue which may not be readily available. Since it is impractical to immunophenotype every sample that is flagged by a complete blood count (CBC) device, passing the specimen through a reliable image-based screening system could potentially reduce cost and patient morbidity.

We designed a texture based solution which distinguishes between normal cells and four different hematologic malignancies. We discriminate among lymphoproliferatie disorders, Chronic Lymphocytic Leukemia (CLL), Mantle Cell Lymphoma, (MCL), Follicular Center Cell Lymphoma (FCC), which can be confused with one another during routine microscopic evaluation. Two acute leukemias, Acute Myelocytic Leukemia (AML) and Acute Lymphocytic Leukemia (ALL), could only be classified in relation to the lymphoproliferatie disorders and normal cells as a single unit labeled as Acute Leukemia. It is shown in [128] that there is no statistical significance in morphometric variables for some subtypes of Acute Leukemia which coincides with our observations. Although each of the disorders under study can exhibit a range of morphological characteristics, Figure 7.1 shows representative morphologies for each.

Chronic Lymphocytic Leukemia (CLL) is the most frequent leukemia in the United States. It is typically a long-term but incurable disease with potential for a more

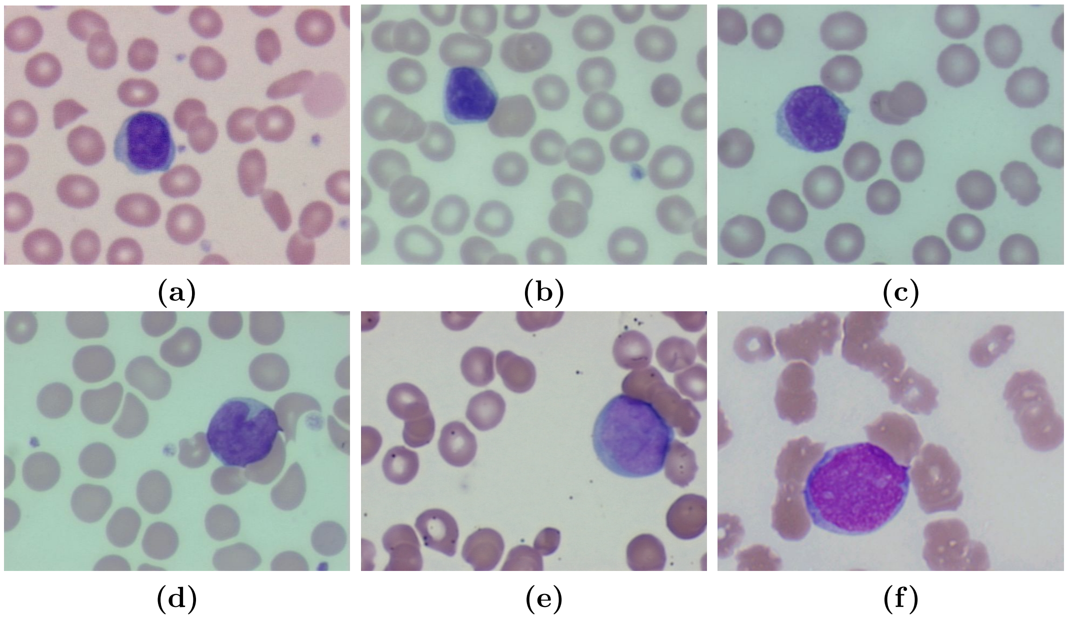


Figure 7.1: Representative morphologies for normal and disorders under study. (a) Normal; (b) Chronic Lymphocytic Leukemia - CLL; (c) Mantle Cell Lymphoma - MCL; (d) Follicular Center Cell Lymphoma - FCC; (e) Acute Myeloblastic Leukemia - AML; (f) Acute Lymphoblastic Leukemia - ALL.

aggressive treatment, e.g. [135]. Mantle Cell Lymphoma (MCL) is a recently described entity (1992) which was not part of the initial working formulation classification system for non-Hodgkin's lymphoma [21]. Timely and accurate diagnosis of MCL is of extreme importance since it has a more aggressive clinical course than CLL or FCC [58, 161]. The third lymphoproliferative disorder under study is Follicular Center Cell Lymphoma (FCC), which is a low-grade lymphoma [2].

Chronic leukemias are associated, at least initially, with well-differentiated, or differentiating leukocytes and a relatively slow course. On the other hand, Acute Leukemias are characterized by the presence of very immature cells (blasts) and by the rapidly fatal course in untreated patients. Acute Leukemia primarily affect adults, with the incidence increasing with age. Despite differences in their cell origin, subtypes of Acute Leukemias share important morphological and clinical features [32]. Our database includes two subtypes of Acute Leukemia, Acute Lymphoblastic Leukemia (ALL) and Acute Myeloblastic Leukemia (AML), but in this study we consider them as a single unit.

The proposed system proceeds in three steps.

1. **Segmentation:** Our observations indicate that both the nucleus and the cytoplasm of a cell contain valuable information regarding the underlying pathology, therefore we analyze them separately. Given a microscopic specimen, initially the system locates the region which contains the cell and then using a robust color gradient vector flow (GVF) active contour model [176] segments the region into nuclear and cytoplasmic components.

2. **Cell representation:** Texture information is used to characterize the morphological structure of normal and diseased cells. A few example images from each disorder are used to create the texton library which captures the structure of texture inside each cells nucleus and cytoplasm. The cells are then represented with two texton histograms corresponding to nuclear and cytoplasmic distribution.

3. **Classification:** We utilize SVMs over the texton histogram based representation to classify the cell images and observe major improvements over the classical histogram based classification methods in the literature such as k-nearest neighbors (kNN).

We conduct four different experiments. In the first experiment, we compare texture features constructed using different methods that were used traditionally in the literature. In the second experiment, we compare the performance of several classification algorithms applied to texture based diagnosis problem. In the third experiment, we show the discriminative power of the proposed cell representation by comparing with several other commonly used features (shape, area) for hematopathological diagnosis. Finally we compare our method with the method of [28] which aims the same problem but has one less disorder class.

7.2 Image Segmentation

In order to extract features from the cell images, we start with the image segmentation. In our application, the region of interest (ROI) containing the object cell, is automatically selected for each image [50]. Both the nuclei and the cytoplasm of cells contain valuable distinguishable information for classification. Therefore, the robust color GVF snake [176], which combines L_2E robust estimation and color gradient, is applied for segmenting both the nuclei and the cytoplasm.

A 2D parametric snake [174] is a curve $\mathbf{x}(s) = (x(s), y(s))$ defined via the parameter $s \in [0, 1]$ to minimize an energy function

$$E_{snake} = \int_0^1 \left(\frac{1}{2} \left(\tau \left| \mathbf{x}'(s) \right|^2 + \rho \left| \mathbf{x}''(s) \right|^2 \right) + E_{ext} \left(\mathbf{x}(s) \right) \right) ds \tag{7.1}$$

where $\mathbf{x}'(s)$ and $\mathbf{x}''(s)$ are the first and second derivatives of the curve with respect to parameter s, and τ and ρ are constants.

According to Helmholtz theorem, the external energy can be replaced with $-\Theta(x, y)$ in (7.1), where Θ is the GVF in image coordinates. The GVF is computed as a diffusion of the gradient vectors of a gray-level edge map derived from the image. The diffusion version of the gradient vector can enlarge the capture region of tradition snake and also lead the snake into concave regions. It is defined as $\Theta(x, y) = [u(x, y), v(x, y)]$ and minimizes the energy function

$$\begin{aligned} \Psi = \quad & \int\int \Big[\eta(u_x^2 + u_y^2 + v_x^2 + v_y^2) + \\ & \left(\left| \nabla \left\{ G_{\sigma(x,y)} * I(x, y) \right\} \right|^2 \cdot \left| \Theta(x, y) - \nabla \left\{ G_{\sigma(x,y)} * I(x, y) \right\} \right|^2 \right) \Big] dxdy \end{aligned} \tag{7.2}$$

where $\nabla \left\{ G_{\sigma(x,y)} * I(x, y) \right\}$ is the gradient of the input image $I(x, y)$ after Gaussian smoothing with covariance $\sigma^2 I_2$ and mean 0; $u_x ... v_y$ are the partial derivatives w.r.t. x and y; η is a constant. In our approach, the original GVF vector field Θ is replaced by Θ^*, which is the diffusion field in the $L^*u^*v^*$ color space that is more closer to the human perception of color. The color differences in this space can be approximated by Euclidean distances [173, Sect. 3.3.9]. Applying the principle of calculus of variations, the problem is solved by the following Euler-Lagrange differential equation:

$$E_{snake} = \tau \mathbf{x}''(s) - \rho \mathbf{x}''''(s) + \Theta^*(x, y) \tag{7.3}$$

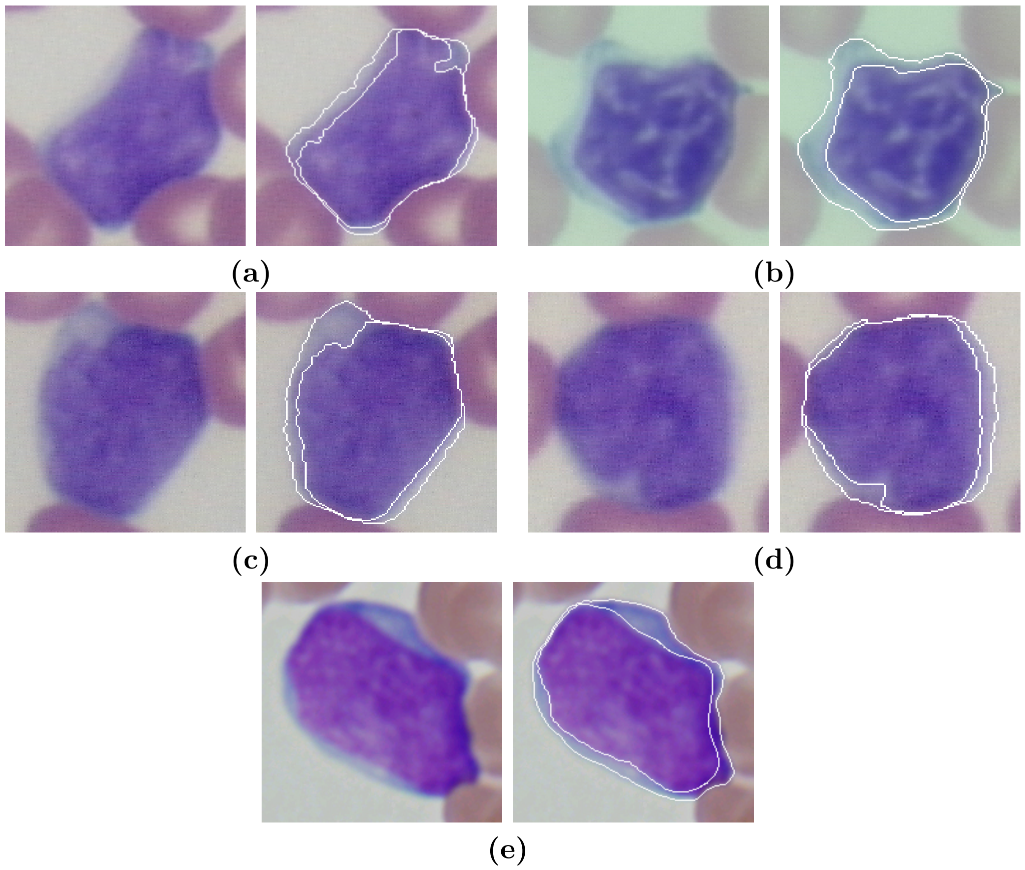


Figure 7.2: Image segmentation results applying robust color GVF snake: (a) Normal; (b) CLL; (c) MCL; (d) FCC; (e) Acute Leukemia. The outer and inner curves correspond to cytoplasm and nucleus segmentations respectively.

where $\mathbf{x}''(s)$ and $\mathbf{x}''''(s)$ are the second and fourth derivatives of the curve with respect to the parameter s. Furthermore, instead of randomly choosing initial curves, we apply L_2E based robust estimation to locate the initial positions, which improves both the convergence speed and the robustness. Figure 7.2 shows some segmentation results using the approach. The performance of segmentation algorithm slightly varies for different classes of disorders and on average 93% of nuclear and 91% of cytoplasmic components are correctly identified. For more details about the color gradient and robust estimation guided active contour segmentation algorithm, we refer to [176].

7.3 Feature Construction

In this section we describe texture based cell representation. A detailed comparison of texture based representation with other common features used in hematapathologic diagnosis is given in Section 7.5.5.

7.3.1 Texture Features

The earlier texture research characterizes a texture according to statistical measures of gray level occurrence relation inside the texture image. The most popular methods are gray level difference [66], gray level cooccurrence matrices [67], gray level run length matrices [57] and autoregressive models [102].

In more recent studies, a texture is characterized through textons which are basic repetitive elements of textures. The form of the textons are not known and they are learned through responses to a set of linear filters, and the resulting responses are clustered. The cluster centers are then selected as the textons. The approach has been successfully used in several fields of texture research including classification, segmentation and synthesis [34, 73, 97, 162].

Recently, it has been proposed that the raw pixel values could replace filter responses to characterize a texture. Local intensity information in spatial domain was used for texture synthesis in [41, 74] and texture classification in [163]. In this approach, the neighboring pixels around the pixel of interest are stacked into an array and used as the feature vector.

After filtering the images, each pixel is mapped to d-dimensional space, where d is the number of filters. Similarly, if local neighborhoods are used, d is equal to size of the local neighborhood. A few sample images are selected from each texture class and the filter responses/local neighborhoods are clustered using k-means clustering algorithm [40]. The cluster centers are selected as textons, therefore the k_m parameter of the clustering algorithm determines the number of textons. This process is repeated for all the c texture classes, and the cluster centers are concatenated to each other to form the texton library. There are $c \cdot k_m = p$ textons in the texton library.

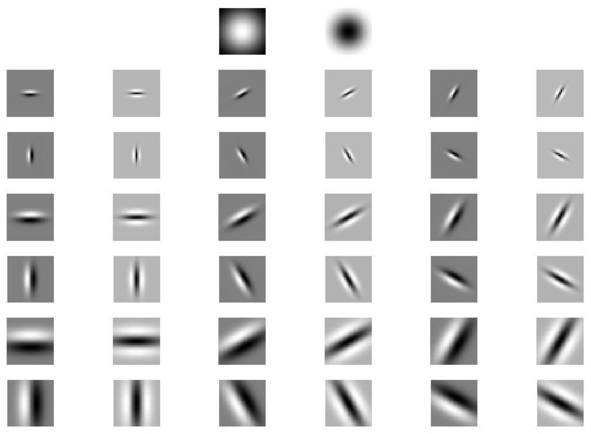


Figure 7.3: **M8** filter bank. There are total of 38 filters from which two filters are rotationally symmetric (Gaussian and Laplacian of Gaussian) and the remaining 36 filters are edge and bar filters at three different scales.

The histograms are created by assigning the filter response of each pixel to the closest texton in the texton library (vector quantization) and then finding the occurrence frequencies of each texton throughout the image. The k-means clustering algorithm used during texton library generation finds a suboptimal solution for determining quantization levels in a d-dimensional space, which is enough in many practical situations. Finally, each texture image is modeled via a p-bin texton histogram, representing all classes.

7.3.2 Cell Representation

The nucleus and cytoplasm of the cells were segmented, as described in Section 7.2. Following segmentation, the cell images are converted to gray level and normalized such that the mean is zero and standard deviation is one. Since the cell images are acquired in different imaging conditions, the normalization is an important operation to minimize the effect of different imaging conditions. The normalization significantly improves the final classification results and supporting claims using similar normalization techniques were also reported in [116]. Notice that, the segmentation algorithm described in previous section uses color information, where as texture representation is computed on normalized gray level images.

We compared the performance of classifiers based on different filter banks and local neighborhood in Section 7.5.3. Although there were not significant differences among some of the filter banks, we found that the **M8** filter bank [162] is the most suitable for our cell representation. The filter bank consists of 38 filters, from which two filters are rotationally symmetric (Gaussian and Laplacian of Gaussian) and 36 of them are edge and bar filters at three scales. We use $\sigma = 10$ for Gaussian and Laplacian of Gaussian functions. The edge and bar filters are selected with $\sigma_x = 1$ and $\sigma_y = 3$ at the finest scale, and are doubled at each of the three scales. The filters are computed at six orientations. Among the oriented filters, only the maximum filter response is retained at each scale, therefore the feature space is $d = 8$ dimensional. The filters in **M8** filter bank are shown in Figure 7.3. The detailed comparison of different filter banks are explained in Section 7.5.3.

We analyze the texture of cytoplasm and nucleus independently. A few random cell images ($n_s = 30$) are selected from each class and filter responses inside the segmentation mask are clustered using k-means clustering algorithm with $k_m = 30$. The clustering is performed separately for pixels inside the nucleus and cytoplasm, and repeated for each disease class. Since the size and the variability of cytoplasm texture is less than the nucleus texture, we generate half the number of clusters from the cytoplasm than from the nucleus. We concatenate the cluster centers from each class and construct the texton libraries separately for cytoplasmic and nuclear texture. The algorithm for texton library generation is given in Figure 7.4.

Using the constructed texton library, the cells are represented with their texton histograms. Given an arbitrary cell image, the pixels inside the cytoplasm and nucleus are filtered and the responses are quantized to the nearest textons in the library. As a result each cell image is represented with two texton histograms of sizes $c \cdot k_m$ and $\frac{c \cdot k_m}{2}$, corresponding to nuclear and cytoplasmic texture. The texton library generation and cell representation process is illustrated in Figure 7.5.

```
Input: Cell image samples ${\mathbf{I}_{j,i}}$, $i = 1...n_s$ from disease classes $j = 1...c$;
cytoplasm and nucleus mask for each cell; number of textons per class $k_m$.

• Initialize nucleus texton library $\mathbf{T}^N = \emptyset$ and cytoplasm texton library
  $\mathbf{T}^C = \emptyset$
• for $j = 1$ to $c$ (for each class)
    – Initialize total filter responses for nucleus $\mathbf{F}^N = \emptyset$ and cytoplasm
      $\mathbf{F}^C = \emptyset$
    – for $i = 1$ to $n_s$ (for each sample in the class)
        ∗ Filter image with M8 filter bank, $\mathbf{F}_{j,i} = \mathbf{f}_{M8} * \mathbf{I}_{j,i}$
        ∗ Concatenate filter responses inside nucleus mask and cytoplasm
          mask to total responses, $\mathbf{F}^N = [\mathbf{F}^N;\ \mathbf{F}^N_{j,i}]$, $\mathbf{F}^C = [\mathbf{F}^C;\ \mathbf{F}^C_{j,i}]$
    – Cluster filter responses for nucleus and cytoplasm
      $\mathbf{T}^N_j = k\text{-means}(\mathbf{F}^N, k_m)$, $\mathbf{T}^C_j = k\text{-means}(\mathbf{F}^C, k_m/2)$
    – Add cluster centers to texton library
      $\mathbf{T}^N = [\mathbf{T}^N;\ \mathbf{T}^N_j]$, $\mathbf{T}^C = [\mathbf{T}^C;\ \mathbf{T}^C_j]$
```

Figure 7.4: Texton library generation

7.4 Classification

We utilize support vector machines (SVMs) to classify among four types of malignancies and normal cells. SVMs were first introduced in [31] for binary classification problems. The technique is a generalization of linear decision boundaries where decision surface is constructed in a large transformed version of the original feature space.

We first focus on the binary classification problem ($c = 2$). Let $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$ be the training set with the respective class labels, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{\pm 1\}$. The SVM solves the following optimization problem

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0,\xi_i} \left[\frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} + \gamma\sum_{i=1}^{n}\xi_i\right] \\ &\text{subject to} \qquad y_i(h(\mathbf{x}_i)^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i \quad \xi_i > 0 \qquad i = 1,..,n \end{aligned} \tag{7.4}$$

where the training samples are mapped to an enlarged space with the function $h(\mathbf{x})$. Minimizing $\boldsymbol{\beta}^T\boldsymbol{\beta}$ is equivalent to maximizing the margin between the positive and negative samples and γ is the tradeoff between the training errors $\{\xi_i\}_{i=1...n}$ and the margin.

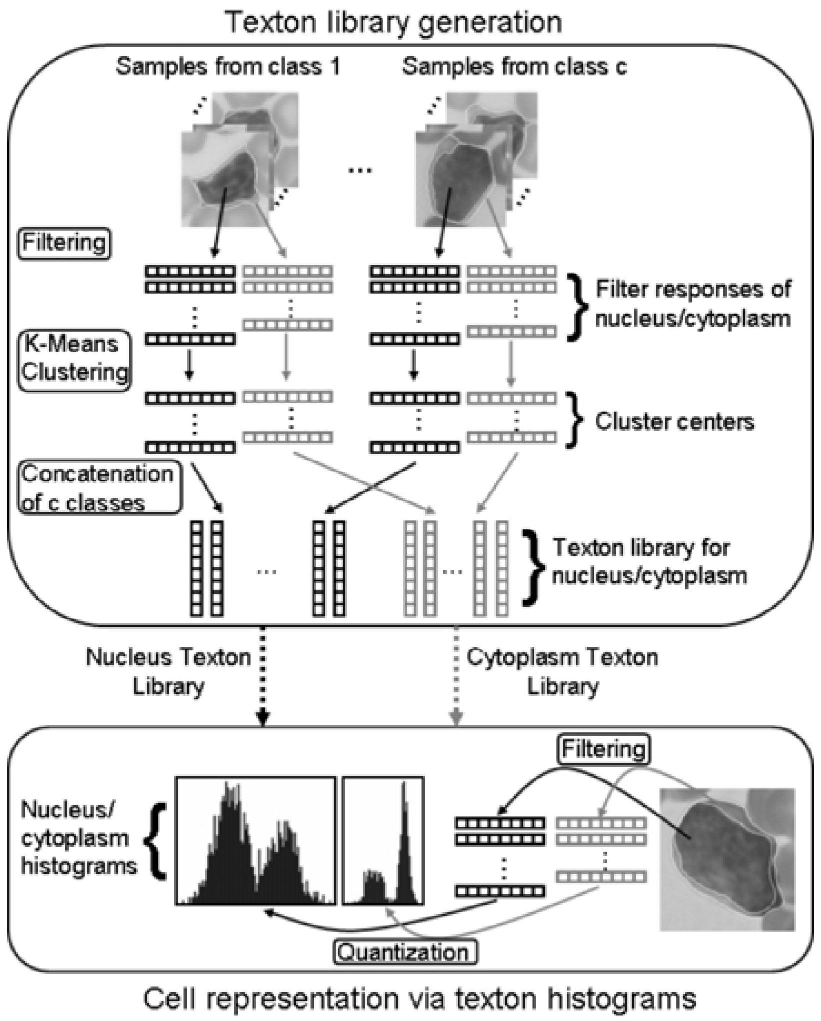

Texton library generation
Samples from class 1
Samples from class c
Filtering
Filter responses of nucleus/cytoplasm
K-Means Clustering
Cluster centers
Concatenation of c classes
Texton library for nucleus/cytoplasm
Nucleus Texton Library
Cytoplasm Texton Library
Filtering
Nucleus/ cytoplasm histograms
Quantization
Cell representation via texton histograms

Figure 7.5: Texton library generation and cell representation. Black and gray vectors correspond to nuclear and cytoplasmic features respectively.

We maximize the dual problem of (7.5) since it is a simpler convex quadratic programming problem. The dual problem and the decision function involve mapping $h(\mathbf{x})$ through an inner product, therefore it suffices to define the inner product through a kernel function without defining the mapping. In our implementation we use the linear kernel function, i.e., $K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$. A more detailed discussion on SVMs can be found in [33].

Next we focus on the multi-class classification problem. The first group of methods construct several binary classifiers and combines them to solve multiclass classification problem. The most popular two methods are one-against-one and one-against-all classifiers. In one-against-one classifier, a binary classifier is trained for all combinations of classes. Then, the label of a test example is predicted by the majority voting among the classifiers. In one-against-all classifier, for each class a binary classifier is trained by labeling the samples from the class as positive examples and samples from the other classes as negative examples. A query point is assigned to the class having maximum decision function among all the classes.

The second group of methods considers all the classes together and solves the multi-class problem in one step. Due to a large scale optimization problem, these methods are computationally more expensive which makes them unsuitable for large size applications. A detailed comparison of multi-class SVMs can be found in [76].

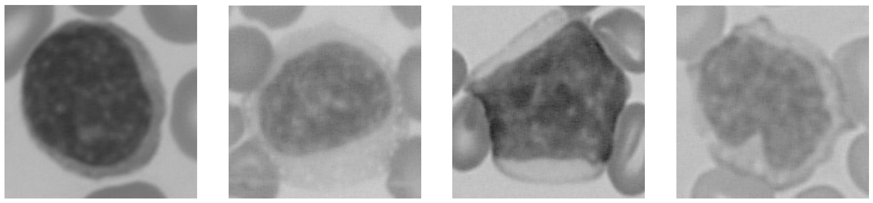
Here we utilize one-against-one SVM classifier. Besides we present results for one-against-all SVM classifier, kNN classifier which is widely used for texture classification problems and LogitBoost classifier which allows us to describe the uncertainty of the classification, in Section 7.5.

7.5 Experiments

7.5.1 Cell Database

Immunophenotyping was used to confirm the diagnosis for a mixed set of 86 hematopathology cases: 18 Mantle Cell Lymphoma (MCL), 20 Chronic Lymphocytic Leukemia (CLL), 9 Follicular Center Cell lymphoma (FCC), and 39 Acute Leukemia. In addition there were 19 normal cases. For each case, we have varying number of cell images ranging from 10 to 90. In total we have 3691 cell images from 105 cases. All cases originated from the archives of either City of Hope National Medical Center in California, University of Pennsylvania of School of Medicine, Spectrum Health System, Grand Rapids, MI or Robert Wood Johnson University Hospital at the University of Medicine & Density of New Jersey.

There were obvious variations in the staining characteristics of specimens among the institutions, which were introduced because of differences in manufacturers of the



(a) RWJ **(b)** PEN **(c)** CAL **(d)** SHS

Figure 7.6: Mantle Cell Lymphoma (MCL) samples from four different institutions. (a) Robert Wood Johnson University Hospital. (b) University of Pennsylvania of School of Medicine. (c) City of Hope National Medical Center in California. (d) Spectrum Health System, Grand Rapids, MI. The images reflect the obvious variations in imaging conditions among different institutions.

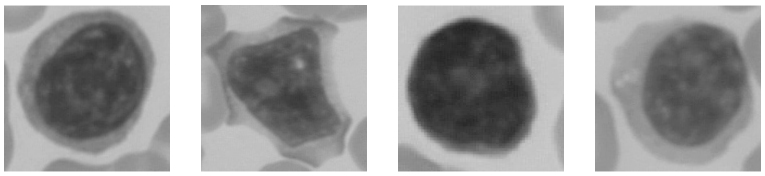
dyes, choices in automated stainers and due to the overall intensity variations. All of these variables led to variations in shadowing, shading, contrasts and highlighting cues providing an added challenge for the classification algorithms. Four MCL samples from different institutions are shown in Figure 7.6, which demonstrates the imaging variations among different institutions.

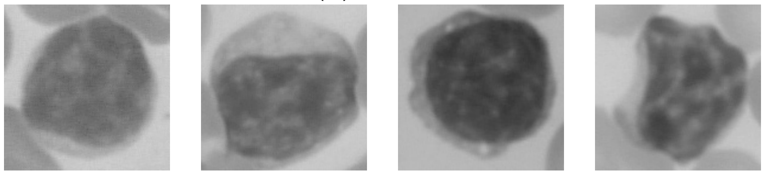
Stained specimens were examined by a certified hematopathologist using an Olympus AX70 microscope equipped with a Prior 6-way robotic stage and motorized turret to locate, digitize and store specimens. The system utilizes interactive software developed in Java and C++. The imaging components of the system consist of an Intel-based workstation interfaced to an Olympus DP70 color camera featuring 12-bit color depth for each color channel and 1.45 million pixel effective resolution. Figure 7.7 shows samples from normal and each disorder category originated from Robert Wood Johnson hospital. As seen in the images our cell database covers a wide range of characteristics for each disease category.

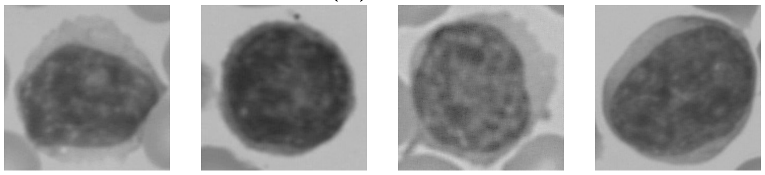
7.5.2 Test Methodology

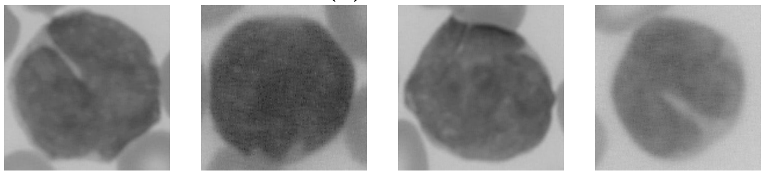
The texture statistics of the cell images from a single case are similar to each other. As a result, the division of test and training sets without obeying the separation of cells based on per case (patient), produce biased results towards better classification rates.

We perform leave one out tests in our experiments. We select all the cell images from a single case as the test set and the remaining cell images in the database as the


(a) Benign


(b) CLL


(c) MCL


(d) FCC

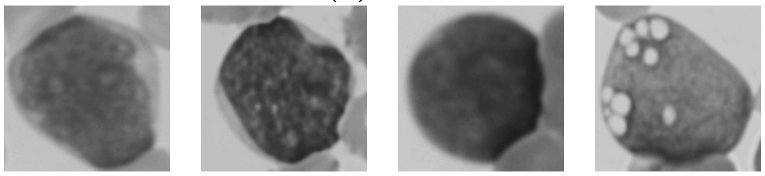

(e) Acute Leukemia

Figure 7.7: Samples from normal and each disorder category from Robert Wood Johnson (RWJ) University Hospital. Even from a single institution, the samples show great variability.

training set. The cells from the selected case are classified using the trained classifier. Training is repeated for each case (patient). In Section 7.5.6, we also present results for 10-fold cross validation, where a model can be trained using some cells of a case and used to classify some other cells of the same case. We refer to this scheme as not obeying the separation.

We present the results for two different tests: cell classification and case classification. In cell classification, we predict the label of each cell with the trained classifier. In the case classification, we assign the label of the case according to the majority voting among its cells. Notice that there are variable number of cell images per case, ranging from 10 to 90.

7.5.3 Filter Banks

In the *first experiment*, we compare the texture features generated by using **M8** filter bank with **LM** [97], **S** [138], **M4** [162] filter banks and the local neighborhood method [163]. The **LM** filter bank consists of 48 anisotropic and isotropic filter: first and second derivative of Gaussians at 6 orientations and 3 scales; 8 Laplacian of Gaussian filters and 4 Gaussian filters. The **S** filter bank consists of 13 rotationally symmetric filters and **M4** filter bank is similar to **M8** filter bank except, edge and bar filters appear only at single scale.

Initially, $n_s = 30$ random cell images are selected from each of the five classes, and the images are convolved with the filter banks. Also, the local neighborhood based features are constructed by stacking 7×7 neighborhood of each pixel. We do not normalize the images while constructing local neighborhood based features since the method uses local intensity information.

For nuclear texture we use $k_m = 30$ cluster centers, and for the cytoplasm texture $k_m = 15$ cluster centers, from each class. Therefore, the texton library has 150 textons for nuclei and 75 textons for the cytoplasms, total of 225 bin histogram for each cell.

The classification performance of different features are given in Table 7.1. The results indicate that **M8**, **S** and **LM** outperforms the **M4** filter bank and local neighborhood method. There are not obvious differences among the **M8**, **S** and **LM** filter banks. The case classification performance of **LM** and **S** filter banks are slightly better than **M8** whereas cell classification performances support the inverse argument. We consider cell classification performance as more important, since cases are classified according the majority voting of the cells and a few classified/misclassified samples among a case can drastically change the result. Moreover, **M8** filter bank is the most compact space which has 8 features whereas **LM** and **S** filter banks have 48 and 13 features, respectively. The clustering and quantization steps take much longer time using the later methods, e.g., one hour for **M8** vs. eight hours for **LM**.

	M8	**S**	**LM**	**M4**	**Local Neigh.**
Cell classification	**84.45**	84.04	82.64	78.98	75.62
Case classification	**89.52**	90.48	90.48	84.76	82.86

Table 7.1: Comparison of **M8** filter bank with features constructed by different filter banks and local neighborhood method. The results are ordered according to cell classification rates.

7.5.4 Classification Methods

In the *second experiment* we compare the one-against-one SVMs with three classification algorithms: LogitBoost, one-against-all SVMs, kNN.

Nearest neighbor classifier with the χ^2 distance metric is the widely applied classification algorithm used with histogram based texture representation. In kNN classification [71, p.415], the closest k_n training samples to the query point are detected and the query point is labeled with the class having the majority votes among the detected points. It is shown with the experiments that among the other possible choices for the distance function (KL-divergence, Bhattacharya distance, Euclidean distance), χ^2 distance performs best for the texture similarity measure. The χ^2 distance between two one-dimensional histograms h_1 and h_2 is measured as

$$\chi^2(h_1, h_2) = \frac{1}{2}\sum_{t=1}^{p}\frac{(h_1(t) - h_2(t))^2}{h_1(t) + h_2(t)}. \tag{7.5}$$

We select the optimum value of the number of neighbors parameter as $k_n = 15$, via cross-validation. Since χ^2 distance measures dissimilarity between two distributions we do not normalize each feature to have zero mean standard deviation for kNN classification, where as we perform normalization for other methods.

The second method, for comparison is multiclass LogitBoost classifier [54]. LogitBoost algorithm learns an additive multiple logistic regression model by minimizing negative log likelihood with quasi-Newton iterations. The probability of a sample $\mathbf{x}$ being in class l is given by

$$p_l(\mathbf{x}) = \frac{e^{F_l(\mathbf{x})}}{\sum_{j=1}^{c} e^{F_j(\mathbf{x})}} \qquad \sum_{j=1}^{c} F_j(\mathbf{x}) = 0 \tag{7.6}$$

where $F_l(\mathbf{x})$ is an additive function

$$F_l(\mathbf{x}) = \sum_{m=1}^{M} f_{ml}(\mathbf{x}). \tag{7.7}$$

At each boosting iteration m, the algorithm learns the weak classifiers $f_{mj}(\mathbf{x}), j = 1...c$ by fitting weighted least squares regressions of training points $\mathbf{x}_i, i = 1...n$ to response values z_{ij} with weights w_{ij} where

$$z_{ij} = \frac{y^*_{ij} - p_j(\mathbf{x}_i)}{p_j(\mathbf{x}_i)(1 - p_j(\mathbf{x}_i))} \qquad w_{ij} = p_j(\mathbf{x}_i)(1 - p_j(\mathbf{x}_i)) \tag{7.8}$$

and y^*_{ij} is the binary class indicator such that $y^*_{ij} = 1$ if class of $i - th$ sample $y_i = j$, and 0 otherwise.

We utilize regression stumps as weak learners, which are regression trees with a single split

$$f(\mathbf{x}) = \begin{cases} a & \text{if } \mathbf{x}^t < \theta \\ b & \text{else.} \end{cases} \tag{7.9}$$

We learn the regression coefficients a, b, the threshold θ while $\mathbf{x}^t$ denotes the $t - th$ dimension among the 225 dimensions of the feature vector $\mathbf{x}$. In our implementation we performed $M = 300$ boosting iterations. We refer readers to [54] for more technical details.

For SVM classifiers we use linear kernel and a soft penalty ($\gamma = 0.01$) for training errors [84, Chapter 11]. The classification rates are given in Table 7.2. Results indicate that we achieve major improvements over the widely used kNN based texture classifier with the introduction of SVMs or LogitBoost. The one-against-one SVM and LogitBoost classifiers produced comparable results, while outperforming the other methods significantly. The performance of one-against-one SVM classifier is better than LogitBoost classifier in cell classification. However, LogitBoost has certain advantages over SVM. In medical applications, it is also important to report the uncertainty about an estimation. Since LogitBoost classifier estimates the posterior distribution of class labels through (7.6), with this method we can describe the uncertainty of the estimation for each individual cell. In our application, we utilize one-against-one SVM classifier since it produces most accurate results.

Our results are comparable to the diagnoses of human experts. In a similar setup, but including fewer cases from normal and three lymphoproliferatie disorders (four class problem), three different human experts could only classify less than 70% of the cells correctly [28], which illustrates the good performance of our approach.

	One-Ag.-One SVM	LogitBoost	One-Ag.-All SVM	kNN
Cell classification	**84.45**	83.14	81.60	81.17
Case classification	**89.52**	89.52	87.62	84.76

Table 7.2: Cell and case classification rates of different classification algorithms.

	Normal	CLL	MCL	FCC	Acute Leukemia
Normal	**734**	64	11	1	0
CLL	35	**504**	49	63	0
MCL	11	78	**375**	27	67
FCC	14	62	31	**132**	2
Acute Leukemia	0	0	59	0	**1372**

Table 7.3: Confusion matrix of cell classification for one-against-one SVM.

In Table 7.3 and Table 7.4 we present the confusion matrices for cell and case classification using one-against-one SVM. The rows of the table show the actual cell classes and the columns show the predicted cell classes. The normal and acute cells are classified accurately, whereas there is some confusion among CLL, MCL and FCC cells. In the case classification almost all of the classes are predicted correctly, and only FCC cases have several misclassifications. This is mainly because we have limited number of training examples from the FCC cases.

Besides five class classification problem, we diagnosed the cells and the cases as normal vs. disorder. Since the problem is binary classification, one-against-one and one-against-all SVMs reduced to binary SVM classifier. The classification rates both for cells and cases are given in Table 7.5. The results indicate that we can diagnose a case as being normal or disorder almost perfectly, and only a single case is misclassified among the whole database.

7.5.5 Other Features

In the *third experiment* we compared the texture based representation with several other features that are commonly used for hematopathology diagnoses. The first set of features are related to area of the cell. We use nucleus and cytoplasm area and nucleus/cytoplasm area ratio.

The second set of features are related to the shape of the nucleus. We analyze the

	Normal	CLL	MCL	FCC	Acute Leukemia
Normal	**19**	0	0	0	0
CLL	0	**18**	1	1	0
MCL	0	2	**14**	0	2
FCC	2	1	1	**5**	0
Acute Leukemia	0	0	1	0	**38**

Table 7.4: Confusion matrix of case classification for one-against-one SVM.

	SVM	LogitBoost	kNN
Cell classification	**98.09**	97.13	96.29
Case classification	**99.05**	99.05	99.05

Table 7.5: Normal vs. disorder classification rates.

shape of the nucleus based on elliptic Fourier descriptors [93] which are made invariant to changes in location, orientation and scale [28]. We achieve rotation invariance by compensating for the arbitrary position of the starting point on the contour and for the arbitrary orientation of the contour. Scale invariance is achieved by normalizing each Fourier coefficient. The following conditions are considered.

- If the first harmonic locus is an ellipse, the rotation is defined relative to the semi-major axis of the locus and we normalize the coefficients by the magnitude of the semi-major axis.
- If the first harmonic locus is circular, the rotation is made with respect to the line defined by the centroid of the contour and the most distant point on the contour and we normalize the coefficients by magnitude of the radius.

We obtain translational invariance by removing the DC coefficient from the Fourier series. We retrieve 16 harmonics (64 coefficients) for the shape of each nucleus.

We present the one-against-one SVM classification results for each of the features and the combination of all the features in Table 7.6. For the combined features we stack all the features into an array. We see that neither area based nor shape based features are alone enough to perform classification. The texture based features outperform both of the other features significantly. Notice that, although indirectly, the area information is presented inside the texture features, since each bin of the texton histogram is equal to the number of occurrence of the texton in the image. There are

	Shape	Area	Texture	Combined
Cell classification	47.43	66.96	84.45	84.62
Case classification	50.47	70.48	89.52	91.42

Table 7.6: Classification rates, based on area, shape, texture and combined features utilizing one-against-one SVMs.

	Normal	CLL	MCL	FCC	Acute Leukemia
Normal	**19**	0	0	0	0
CLL	0	**18**	1	1	0
MCL	0	2	**14**	0	2
FCC	1	0	1	**7**	0
Acute Leukemia	0	0	1	0	**38**

Table 7.7: Confusion matrix of case classification using combined features utilizing one-against-one SVMs.

minor improvements from texture based representation, 84.45%, to combined features, 84.62%, in cell classification and 89.52% to 91.42% in case classification.

The distribution of the classification performances according to different disorders show variation from texture based features to combined features. The confusion matrix of case classification using combined features are given in Table 7.7. Usually in the advanced stages of FCC, the nuclei show variability from the other diseases. We see that two more FCC cases are correctly classified with combined features relative to Table 7.4. The results almost did not affect the other classes which supports the claim. We achieve only minor improvements over the texture features with the introduction of morphological features such as area and shape.

7.5.6 Comparison with Previous Method

In this section we compare our approach with the method of [28]. The problem considered in [28] is a subset of our problem, where only four classes are considered (Benign, MCL, FCC, CLL). The cell database of [28] contains only 261 specimens and the testing is performed by adopting 10-fold cross validations which do not obey separation based on patient.

The results of [28] is given in Table 7.8. To directly compare our results with [28] and illustrate the effect of test methodology, we also performed 10-fold cross validations

	Normal	CLL	MCL	FCC	No Decision
Normal	**73.0**	13.4	0.0	12.0	1.6
CLL	7.0	**83.9**	7.1	2.0	0
MCL	0	13.6	**83.3**	1.4	1.7
FCC	5.0	2.5	0.0	**90.0**	2.5

Table 7.8: Confusion matrix of cell classification rates of [28].

	Normal	CLL	MCL	FCC	AML
Normal	**96.2**	3.4	0.4	0.0	0.0
CLL	2.9	**90.4**	3.9	2.8	0
MCL	1.5	6.0	**83.6**	1.5	7.4
FCC	1.9	9.7	6.2	**81.4**	0.8
AML	0.0	0.0	1.4	0.0	**98.9**

Table 7.9: Confusion matrix of cell classification rates using 10-fold cross validation utilizing one-against-one SVMs.

and presented the cell classification results in Table 7.9. Even though the problem that we solve is more difficult (one more class), we see that our results are significantly better than [28] except for FCC class. Only the classification of FCC cells were slightly better in [28], but we note that there were only 20 FCC cells in their experiments.

The classification rates of our method with 10-fold cross validation tests are significantly higher than the leave-one-out tests performed previously. The cell classification rate changed from 84.45% to 93.18%. We see that the separation of training and test sets without obeying case separation, produces biased results towards better performances. The specimens from a single case may have similarities to each other, which are uncorrelated to the class of the disorder.

Chapter 8

Conclusions

This thesis proposed and investigated novel learning methods which account for the manifold structure of the visual data. Three learning algorithms are presented based on the problem setting: Unsupervised classification (clustering), supervised classification and regression.

We presented an unsupervised learning algorithm by extending the application domain of the mean shift algorithm from vector spaces to Lie groups. The derived clustering algorithm on Lie groups is used for multiple motion estimation problem from noisy point correspondences. The problem is considered in its most general form such that the number of motions in the scene is not known priory and there exist large amount of outliers.

In a supervised setting, we described an additive classification model for data points lying on a Riemannian manifold. The derived algorithm is applied to pedestrian detection problem which is known to be among the hardest examples of the detection tasks. Extensive experiments on two challenging human datasets show that the algorithm is superior to the current state of the art detection approaches and the vector space methods.

Next, we presented a regression model where the response parameters form a Lie group. A novel formulation for affine tracking is derived by learning the regression model on the Lie algebra of the affine group, given the image observations. The approach is shown to yield significantly lower estimation errors compared with the vector space methods. The learning model is generalized to build an invariant object detector from a pose dependent detector. Compared to the existing detection methods, the size of the search space is reduced drastically.

Our main observation is that accounting for the non-Euclidean nature of the visual data provides major improvements over the vector space approaches. Several other contributions of the thesis include a novel region descriptor and an online learning algorithm for modeling background statistics of a scene. The proposed methods are utilized for for several challenging problems such as matching, tracking, texture classification and low frame rate tracking.

8.1 Directions for Future Research

The following are a few research directions that can be pursued based on the techniques proposed in this thesis.

8.1.1 Incremental Subspace Estimation

The bottleneck of the presented mean shift procedure is the sampling step required to generate hypothesis from the available point set. The number of elemental subsets required to correctly estimate the parameters is an exponential function of the fraction of inlier points and the cardinality of the elemental subsets. The probability of getting a single good estimate from elemental subsets can be formulated as $\left(\frac{n_i}{n_i+n_o}\right)^d$, where n_i and n_o are the number of inlier and outlier points respectively and d is the cardinality of the elemental subsets. The probability can be quite small if the cardinality is large or fraction of the inliers is small. For example, in fundamental matrix estimation problem the cardinality of the elemental subsets is eight and the fraction of inliers can be quite small since they are acquired from the output of a point matching procedure.

The proposed mean shift algorithm can be modified to perform incremental subspace estimation where a solution can be found in polynomial time as a parameter of the number of total points and the cardinality. The idea is to enforce lower rank constraints and increment the constraint by one at each iteration. The sketch of the approach is as follows. Assuming the points lie on a k-dimensional subspace of an m-dimensional space, the algorithm initially searches for a one-dimensional subspace which is contained in the the k-dimensional subspace and does not have any component in the null space.

At each iteration one more basis vector is added to the estimated subspace until the k-dimensional subspace is recovered. The operation briefly outlined here can be performed by clustering lower dimensional subspaces on Grassmann manifold and adding basis vectors which are orthogonal to the current estimates.

8.1.2 Multiclass Classification and Applications to Other Manifolds in Vision

The classification algorithm presented in this thesis is designed for binary classification problems. The approach can be trivially generalized to multiclass setting by solving for multi-logit parametrization [54].

There exist many different manifolds which are well known outside vision and have been studied in fields such as physics and robotics. These spaces can be further analyzed for practical applications in computer vision problems based on the learning techniques presented in this thesis. For example, the presented supervised learning algorithm can be utilized for classification on Grassmann manifold for shape space analysis [8] or Special Euclidean Group for motion estimation.

8.1.3 Clustering Point Sets

Many vision problems involve estimation on variable length spaces where the data points are unordered. For instance, the bag of features model is a commonly used representation model where typically an image or an object of interest is represented with several local features extracted from the source. Since many of the classical machine learning techniques require the features to be ordered and fixed length, they are not well suited for these spaces. The common approach is to present an auxiliary representation where the induced space is fixed length and the points are ordered, such as the occurrence frequencies of the local features in the form of an histogram.

The mean shift procedure can be generalized for direct analysis of such feature spaces without requiring an auxiliary representation. The problem can be formulated as clustering of point sets, where the cardinality of the sets are not constant and the points are unordered. Initially, the method requires solving an assignment problem among two sets based on weighted bipartite graph matching, where the weights are adjusted through the matching scores among the unordered point pairs. Then the mean shift vector can be defined on a lower dimensional space using only the matched points. It can be shown that the procedure is convergent to a local maxima of an appropriately defined kernel density on the space of point sets.

8.1.4 Joint motion and image segmentation

It is possible to extend the multiple motion estimation framework for detecting multiple structures from uncalibrated image sequences. The approach initially detects the optical flow field between image sequences. In small neighborhoods of the image, it is possible to assume a simple motion model such as translational motion which can be acquired from the optical flow field. In larger neighborhoods a more complex motion model is needed. Starting from simple translational motion, the model can be generalized to hold for larger regions such as two-dimensional rigid motion to affine to planer homography. The segmentation of the scene and the motion parameters can be acquired simultaneously by incremental clustering.

Appendix A

Riemannian Geometry Overview

Here we present a brief introduction to Riemannian geometry. For further information, we refer readers to textbooks on the subject [14, 85].

A.1 Topological Spaces

A *topological space* is a set S together with a class of open subsets $\mathcal{T}$ of S. The $(S, \mathcal{T})$ pair satisfies following four axioms:

- The empty set is in $\mathcal{T}$.
- The set S is in $\mathcal{T}$.
- The union of any collection of sets in $\mathcal{T}$ is in $\mathcal{T}$.
- The intersection of any finite number of sets in $\mathcal{T}$ is in $\mathcal{T}$.

The elements of set S are called the points and the open subsets $\mathcal{T}$ is called the topology of S. Any open set $\mathcal{U} \in \mathcal{T}$ which contains point $\mathbf{X} \in S$ is called the neighborhood of the point. A topological space is called *Hausdorff* if any two points can be separated by neighborhoods, such that, $\mathbf{X}, \mathbf{Y} \in S$ and there exists $\mathcal{U}, \mathcal{V} \in \mathcal{T}$, $\mathbf{X} \in \mathcal{U}$, $\mathbf{Y} \in \mathcal{V}$ and $\mathcal{U} \cap \mathcal{V} = \emptyset$. A mapping between two topological spaces is called *continuous* if the inverse image of any open set with respect to the mapping is again an open set. A bijective (one-to-one and onto) mapping which is also continuous in both directions is called a *homeomorphism*.

A.2 Smooth Manifolds

An m-dimensional manifold $\mathcal{M}$ is a topological space which is locally similar to $\mathbb{R}^m$. More formally, for any point $\mathbf{X} \in \mathcal{M}$, there exist an open neighborhood $\mathcal{U} \subset \mathcal{M}$ containing the point and a homeomorphism ϕ mapping the neighborhood to an open set $\mathcal{V} \subset \mathbb{R}^m$, such that, $\phi : \mathcal{U} \mapsto \mathcal{V}$. The mapping can be written as $\phi(\mathbf{X}) = [\phi^1(\mathbf{X}) \;\; \phi^2(\mathbf{X}) \;\; \ldots \;\; \phi^m(\mathbf{X})]$ where $\phi^i, i = 1 \ldots m$ are called the *coordinate functions.* We call the pair $(\mathcal{U}, \phi)$ a *coordinate chart.* An *atlas* is a collection of charts $\{\mathcal{U}_\alpha, \phi_\alpha\}$ for which the $\mathcal{U}_\alpha$ form an open covering of $\mathcal{M}$, such that, $\bigcup_\alpha \mathcal{U}_\alpha = \mathcal{M}$.

Let $(\mathcal{U}_\alpha, \phi_\alpha)$ and $(\mathcal{U}_\beta, \phi_\beta)$ be two coordinate charts, such that, $\mathcal{U}_\alpha \cap \mathcal{U}_\beta$ is non empty. The chart transition $\phi_\alpha \circ \phi_\beta^{-1}$ is a mapping between two open sets in $\mathbb{R}^m$, such that,

$$\phi_\alpha \circ \phi_\beta^{-1} : \phi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \mapsto \phi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta). \tag{A.1}$$

If all the chart transitions are smooth functions the atlas is called a smooth structure. An m-dimensional smooth manifold is a manifold of dimensional m with a smooth structure.

Let $f : \mathcal{M} \mapsto \mathbb{R}$ be a real valued function on a manifold. Given a coordinate chart $(\mathcal{U}, \phi)$, $f \circ \phi^{-1}$ is a mapping from $\mathbb{R}^m$ to $\mathbb{R}$. The function f is called smooth if for all coordinate charts the mapping is a smooth map from $\mathbb{R}^m$ to $\mathbb{R}$. Similarly a mapping $f : \mathcal{M} \mapsto \mathcal{M}'$ between smooth manifolds with charts $(\mathcal{U}_\alpha, \phi_\alpha)$ and $(\mathcal{U}'_\alpha, \phi'_\alpha)$ is called smooth if for all charts $\phi'_\alpha \circ f \circ \phi_\alpha^{-1}$ is a smooth map.

A.3 Tangent Spaces

Let $\mathcal{M}$ be an m-dimensional smooth manifold, I an open interval in $\mathbb{R}$ and $\gamma(t) : I \mapsto \mathcal{M}$ be a curve. We say that γ is a smooth curve if for each chart $(\mathcal{U}_\alpha, \phi_\alpha)$, $\phi_\alpha \circ \gamma : \mathbb{R} \mapsto \mathbb{R}^m$ is a smooth map where $\gamma(I) \cap \mathcal{U}_\alpha \neq \emptyset$.

Suppose two smooth curves $\gamma_1 : (-\epsilon, \epsilon) \mapsto \mathcal{M}$ and $\gamma_2 : (-\epsilon, \epsilon) \mapsto \mathcal{M}$ with $\gamma_1(0) = \gamma_2(0) = \mathbf{X}$. The curves are called equivalent if the ordinary derivatives of $\phi_\alpha \circ \gamma_1$ and $\phi_\alpha \circ \gamma_2$ at 0 coincide for all charts $(\mathcal{U}_\alpha, \phi_\alpha)$ where $\mathbf{X} \in \mathcal{U}_\alpha$. A *tangent vector* at $\mathbf{X}$ is defined by the equivalence class of the smooth curves $\gamma : (-\epsilon, \epsilon) \mapsto \mathcal{M}, \gamma(0) = \mathbf{X}$. The tangent space at $\mathbf{X}$, $T_{\mathbf{X}}M$, is the set of all tangent vectors at $\mathbf{X}$.

The definition provides a simple geometric interpretation of the tangent space. Tangent vectors are the tangents to the smooth curves lying on the manifold. However, there is no single representative curve for a given tangent vector.

The tangent space is a vector space, thereby it is closed under addition and scalar multiplication. Suppose γ_1 and γ_2 are two smooth curves on $\mathcal{M}$ where $\gamma_1(0) = \gamma_2(0) = \mathbf{X}$. We can not directly generate new tangent vectors by simply adding or multiplying the curves since the resulting curve is not necessarily contained in $\mathcal{M}$. However, we can always choose a chart $(\mathcal{U}_\alpha, \phi_\alpha)$, $\mathbf{X} \in \mathcal{U}_\alpha$, such that $\phi_\alpha(\mathbf{X}) = 0$. The mappings $\phi_\alpha(\gamma_1(t))$ and $\phi_\alpha(\gamma_2(t))$ define two curves through the origin in the coordinate chart. Now we can define new curves on the coordinate space and pull back the results via ϕ_α^{-1}

$$\gamma_s(t) = \phi_\alpha^{-1}\left(\phi_\alpha(\gamma_1(t)) + \phi_\alpha(\gamma_2(t))\right) \tag{A.2}$$

and

$$\gamma_m(t) = \phi_\alpha^{-1}\left(\lambda\phi_\alpha(\gamma_1(t))\right). \tag{A.3}$$

It can be easily verified that the pull back curves are contained in $\mathcal{M}$ for $t \in (-\epsilon, \epsilon)$ and $\gamma_s(0) = \gamma_m(0) = \mathbf{X}$. By chain rule, their derivatives at $t = 0$ generate the sum and the scalar multiples of the corresponding tangent vectors.

A natural basis for the tangent space $T_{\mathbf{X}}M$ can be inherited through coordinate charts. Let $(\mathcal{U}_\alpha, \phi_\alpha)$ be a coordinate chart containing point $\mathbf{X}$. Now we can consider the curves

$$\gamma^j(t) = \begin{cases} t + const, & i = j; \\ const, & i \neq j. \end{cases} \tag{A.4}$$

in terms of the local coordinates $u^i, i = 1 \ldots m$. The image of the curves on $\mathcal{M}$ can be computed through $\phi_\alpha^{-1}(\gamma^j)$ and we can always select the constants such that $\phi_\alpha^{-1}(\gamma^j(0)) = \mathbf{X}$. The associated tangent vector at $t = 0$ is called $\frac{\partial}{\partial \mathbf{x}^i}$ and has the local coordinates

$$v^j = \frac{du^j}{dt}\Big|_{t=0} = \begin{cases} 1, & i = j; \\ 0, & i \neq j. \end{cases} \tag{A.5}$$

The global coordinates can be recovered by the chain rule

$$w^i = \sum_{j=1}^{m} \frac{\partial \left((\phi_\alpha^{-1})^i\right)}{\partial u^j} v^j. \tag{A.6}$$

A.4 Riemannian Manifolds

A symmetric positive definite bilinear form on a vector space T is a map $< .,. >$: $T \times T \mapsto \mathbb{R}$ which is linear in both parameters, symmetric and positive definite. Let $\mathbf{v}, \mathbf{w}, \mathbf{z} \in T$ and $\alpha, \beta \in \mathbb{R}$, then $< .,. >$ satisfies

- *Bilinearity:* $< \alpha\mathbf{v} + \beta\mathbf{w}, \mathbf{z} >= \alpha < \mathbf{v}, \mathbf{z} > +\beta < \mathbf{w}, \mathbf{z} >$
  $< \mathbf{z}, \alpha\mathbf{v} + \beta\mathbf{w} >= \alpha < \mathbf{z}, \mathbf{v} > +\beta < \mathbf{z}, \mathbf{w} >$
- *Symmetry:* $< \mathbf{v}, \mathbf{w} >=< \mathbf{w}, \mathbf{v} >$
- *Positive Definiteness:* $< \mathbf{v}, \mathbf{v} >\geq 0$ with equality occurring if and only if $\mathbf{v} = 0$.

A smooth Riemannian metric on a manifold $\mathcal{M}$ is an association to every point $\mathbf{X} \in \mathcal{M}$ a symmetric positive definite bilinear form $< .,. >_{\mathbf{X}}$: $T_{\mathbf{X}}M \times T_{\mathbf{X}}M \mapsto \mathbb{R}$ which depends smoothly on the base point. The metric induces a norm for tangent vectors in the tangent space, such that, $\mathbf{v} \in T_{\mathbf{X}}M$, $\|\mathbf{v}\|_{\mathbf{X}}^2 =< \mathbf{v}, \mathbf{v} >_{\mathbf{X}}$. A Riemannian manifold is a smooth manifold equipped with a smooth Riemannian metric $(\mathcal{M}, < .,. >_{\mathbf{X}})$. Note that, it is possible to define different metrics on the same manifold and different Riemannian manifolds can be obtained.

Given a coordinate chart $(\mathcal{U}_\alpha, \phi_\alpha)$, we can take the bases of the tangent space to be the coordinate bases $\frac{\partial}{\partial \mathbf{x}^i}$ as defined before. The metric can be expressed by an $m \times m$ positive definite symmetric matrix where the ij-th entry is given by

$$g(\mathbf{X})_{i,j} =< \frac{\partial}{\partial \mathbf{x}^i}, \frac{\partial}{\partial \mathbf{x}^j} >, i, j = 1 \ldots m. \tag{A.7}$$

The inner product of two tangent vectors $\mathbf{v} = \sum_{i=1}^m v^i \frac{\partial}{\partial \mathbf{x}^i}, \mathbf{w} = \sum_{i=1}^m w^i \frac{\partial}{\partial \mathbf{x}^i}$ is then given by

$$< \mathbf{v}, \mathbf{w} >_{\mathbf{X}}= \sum_{i,j=1}^{m} g(\mathbf{X})_{i,j} v^i w^j. \tag{A.8}$$

A.5 Geodesics

Let $\gamma(t) : [t_0, t_1] \mapsto \mathcal{M}$ be a smooth curve on $\mathcal{M}$. The length of the curve $L(\gamma)$ is defined as

$$L(\gamma) = \int_{t_0}^{t_1} \|\gamma'(t)\|_{\gamma(t)} dt. \tag{A.9}$$

A smooth curve is called *geodesic* if and only if its velocity vector is constant along the curve, such that, $\|\gamma'(t)\|_{\gamma(t)} = const$ for $t \in [t_0, t_1]$.

Let $\mathbf{X}$ and $\mathbf{Y}$ be two points on $\mathcal{M}$. The distance between the points, $d(\mathbf{X}, \mathbf{Y})$, is the infimum of the length of the curves, such that, $\gamma(t_0) = \mathbf{X}$ and $\gamma(t_1) = \mathbf{Y}$. All the shortest length curves between the points are geodesics but not vice-versa. However, for nearby points the definition of geodesic and the shortest length curve coincide. Throughout the thesis we use geodesics and the shortest length curves interchangeably.

A.6 Exponential Maps

Let $\mathbf{X}$ be a point on $\mathcal{M}$. For each tangent vector $\mathbf{v} \in T_{\mathbf{X}}M$, there exists a unique geodesic γ starting at $\gamma(0) = \mathbf{X}$ having initial velocity $\gamma'(0) = \mathbf{v}$. The *exponential map*, $\exp_{\mathbf{X}} : T_{\mathbf{X}}M \mapsto \mathcal{M}$, maps the tangent $\mathbf{v}$ to the point on the manifold reached by the geodesic after unit time

$$\exp_{\mathbf{X}}(\mathbf{v}) = \gamma(1). \tag{A.10}$$

Since the velocity along the geodesic is constant, the length of the geodesic is given by the norm of the initial velocity $d(X, \exp_{\mathbf{X}}(\mathbf{v})) = \|\mathbf{v}\|_{\mathbf{X}}$. Under the exponential map, the image of the zero tangent vector is the point itself, $\exp_{\mathbf{X}}(0) = \mathbf{X}$.

For each point on the manifold $\mathbf{X} \in \mathcal{M}$, the exponential map, $\exp_{\mathbf{X}}$, is a diffeomorphism (one-to-one, onto and continuously differentiable mapping in both directions) from a neighborhood of the origin of the tangent space $T_{\mathbf{X}}M$ onto a neighborhood of the point $\mathbf{X}$. Within this neighborhood, the inverse mapping, $\log_{\mathbf{X}} : \mathcal{M} \mapsto T_{\mathbf{X}}M$, is uniquely defined. For certain manifolds the neighborhoods can be extended to the whole tangent space and manifold hence the exponential map is a global diffeomorphism. Let $\mathbf{X}, \mathbf{Y} \in \mathcal{M}$. From the definition of geodesic and the exponential map, the distance between the points can be computed by

$$d(\mathbf{X},\mathbf{Y}) =< \log_{\mathbf{X}}(\mathbf{Y}), \log_{\mathbf{X}}(\mathbf{Y}) >_{\mathbf{X}} = \|\log_{\mathbf{X}}(\mathbf{Y})\|_{\mathbf{X}}. \tag{A.11}$$

A.7 Types of Riemannian Manifolds

Here we briefly describe the geometry of a few examples of Riemannian manifolds on which we derive our applications.

A.7.1 Lie Groups

A group G is a set of elements with an associated group operation (multiplication) that satisfy four axioms:

- *Closure:* The group is closed under group operation. $\mathbf{X} \in G$ and $\mathbf{Y} \in G$ implies $\mathbf{XY} \in G$
- *Associativity:* The group operation is associative. $\mathbf{X}(\mathbf{YZ}) = (\mathbf{XY})\mathbf{Z}$
- *Identity:* There is an identity element $\mathbf{I}$ in the group. $\mathbf{IX} = \mathbf{XI} = \mathbf{X}$
- *Inverse:* There is an inverse for each element in the group. $\mathbf{XX}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$.

In addition, a group is called commutative if the group operation is commutative, such that $\mathbf{XY} = \mathbf{YX}$.

A Lie group is a group G with the structure of a smooth manifold such that the group operations are analytic, i.e. the maps

$$G \times G \rightarrow G, \quad (\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{XY} \quad \textit{and} \tag{A.12}$$

$$G \rightarrow G, \quad \mathbf{X} \rightarrow \mathbf{X}^{-1} \tag{A.13}$$

are analytic [133].

The tangent space to the identity element of the group $T_{\mathbf{I}}G$ forms a Lie algebra which is denoted by $\mathfrak{g}$. A Lie algebra $\mathfrak{g}$ is a vector space that is closed under the Lie bracket operation:

$$\mathbf{x}, \mathbf{y} \in \mathfrak{g} \quad \textit{implies} \quad [\mathbf{x}, \mathbf{y}] \in \mathfrak{g}. \tag{A.14}$$

A Lie bracket is a bilinear operation that satisfies the following identities:

- *Anti-symmetry:* $[\mathbf{x},\mathbf{y}] = -[\mathbf{y},\mathbf{x}]$
- *Jacobi identity:* $[\mathbf{x},[\mathbf{y},\mathbf{z}]] + [\mathbf{y},[\mathbf{z},\mathbf{x}]] + [\mathbf{z},[\mathbf{x},\mathbf{y}]] = 0$

The group operation provides Lie groups with additional algebraic structure. Let $\mathbf{X} \in G$. Left multiplication by the inverse of the group element

$$\mathbf{X}^{-1} : G \to G \tag{A.15}$$

maps the neighborhood of $\mathbf{X}$ to neighborhood of $\mathbf{I}$ and carries the inner product at $T_{\mathbf{X}}G$, to the Lie algebra $\mathfrak{g}$. The inverse mapping is defined by left multiplication by $\mathbf{X}$. Using this mapping we can work on the Lie algebra instead of $T_{\mathbf{X}}G$. For instance we only need to define the exponential map at the identity element, $\exp : \mathfrak{g} \to G$ which is a diffeomorphism from a neighborhood of the origin of the Lie algebra $\mathfrak{g}$ onto a neighborhood of the identity $\mathbf{I}$.

For noncommutative Lie groups the identity $\exp(\mathbf{x})\exp(\mathbf{y}) = \exp(\mathbf{x}+\mathbf{y})$ does not hold. The identity is expressed by Baker-Campbell-Hausdorff formula [133, p.22-23] $\exp(\mathbf{x})\exp(\mathbf{y}) = \exp(\mathrm{BCH}(\mathbf{x},\mathbf{y}))$ where

$$\mathrm{BCH}(\mathbf{x},\mathbf{y}) = \mathbf{x} + \mathbf{y} + \frac{1}{2}[\mathbf{x},\mathbf{y}] + O(|(\mathbf{x},\mathbf{y})|^3). \tag{A.16}$$

Matrix Lie groups are all the subgroups of the general linear group $\mathbf{GL}(d,\mathbb{R})$ which is the group of $d \times d$ dimensional real nonsingular matrices. The group operation, matrix multiplication, is associative and every nonsingular matrix has an inverse. The Lie bracket operator is defined as

$$[\mathbf{x},\mathbf{y}] = \mathbf{xy} - \mathbf{yx}. \tag{A.17}$$

Matrix groups are probably the most well known examples of Lie groups. The exponential map of a matrix is given by the ordinary matrix exponential

$$\exp(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{x}^k. \tag{A.18}$$

The inverse map

$$\log(\mathbf{X}) = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k}(\mathbf{X}-\mathbf{I})^k \tag{A.19}$$

can only be defined on a neighborhood of $\mathbf{I}$. When $\mathbf{X}$ is distant from the identity element of the group, the series fails to converge.

Let $\mathbf{X}, \mathbf{Y} \in G$ where G is a matrix Lie group. Using the definition of geodesics and the mapping (A.15), the distance function is given by

$$d(\mathbf{X}, \mathbf{Y}) = \|\log(\mathbf{X}^{-1}\mathbf{Y})\|_F \tag{A.20}$$

where $\|.\|_F$ denotes the Frobenius norm of a matrix.

Special Orthogonal Group

The special orthogonal group $\mathbf{SO}(3)$ is the group of rotations $\mathbf{R}$ in 3D. The rotation group satisfies $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\det(\mathbf{R}) = 1$.

Its associated Lie algebra $\mathfrak{so}(3)$ is the set of 3×3 skew-symmetric matrices

$$\boldsymbol{\Omega} = \begin{pmatrix} 0 & -\omega_x & \omega_y \\ \omega_x & 0 & -\omega_z \\ -\omega_y & \omega_z & 0 \end{pmatrix}. \tag{A.21}$$

We can write skew-symmetric matrices in vector form $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ and the identity $\boldsymbol{\Omega}\mathbf{x} = \boldsymbol{\omega} \times \mathbf{x}$ always holds. From a geometrical point of view, $\boldsymbol{\Omega}$ can be considered as rotation around axis $\boldsymbol{\omega}/\|\boldsymbol{\omega}\|$ by an angle $\|\boldsymbol{\omega}\|$. The structure of $\mathbf{SO}(3)$ allows us to compute exponential map $\mathfrak{so}(3) \to \mathbf{SO}(3)$ analytically via the Rodrigue's rotation formula [88, p.204]

$$\exp(\boldsymbol{\Omega}) = \mathbf{I} + \frac{\sin\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\boldsymbol{\Omega} + \frac{1-\cos\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^2}\boldsymbol{\Omega}^2. \tag{A.22}$$

The inverse map $\log(\mathbf{R})$ can be found in two steps [141, p.51]

$$\cos\theta = \frac{1 - tr(\mathbf{R})}{2} \tag{A.23}$$

and

$$\log(\mathbf{R}) = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^T). \tag{A.24}$$

The method fails if $\theta = \pi$, since $\sin\pi = 0$.

Special Euclidean Group

The special Euclidean group $\mathbf{SE}(3)$ is the group of rigid motions in 3D

$$\mathbf{M} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{A.25}$$

where rotation matrix $\mathbf{R}$ is in $\mathbf{SO}(3)$ and translation vector $\mathbf{t}$ is in $\mathbb{R}^3$.

The Lie algebra of the rigid motions $\mathfrak{se}(3)$ are the set of matrices

$$\mathfrak{m} = \begin{pmatrix} \mathbf{\Omega} & \mathbf{u} \\ \mathbf{0}^T & 0 \end{pmatrix} \tag{A.26}$$

where $\mathbf{u}$ is in $\mathbb{R}^3$ and $\mathbf{\Omega}$ is defined in (A.21). The analytical computation of exponential and logarithm maps are very similar to $\mathbf{SO}(3)$ and can be found in [141, p.52].

Affine Group

A two-dimensional affine transformation is given by a 3×3 matrix $\mathbf{M}$

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & 1 \end{pmatrix} \tag{A.27}$$

where $\mathbf{A}$ is a nonsingular 2×2 matrix and $\mathbf{b} \in \mathbb{R}^2$. The set of all two-dimensional affine transformations forms a matrix Lie group denoted by $A(2)$.

The associated Lie algebra is the set of matrices

$$\mathbf{m} = \begin{pmatrix} \mathbf{U} & \mathbf{v} \\ 0 & 0 \end{pmatrix} \tag{A.28}$$

where, $\mathbf{U}$ is a 2×2 matrix and $\mathbf{v} \in \mathbb{R}^2$.

A.7.2 Space of Symmetric Positive Definite Matrices

The $d \times d$ dimensional symmetric positive definite matrices (nonsingular covariance matrices), Sym_d^+, can be formulated as a connected Riemannian manifold and an affine invariant Riemannian metric on the tangent space of Sym_d^+ is given by [122]

$$< \mathbf{y}, \mathbf{z} >_{\mathbf{X}} = \text{trace}\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{y}\mathbf{X}^{-1}\mathbf{z}\mathbf{X}^{-\frac{1}{2}}\right). \tag{A.29}$$

The exponential map associated to the Riemannian metric

$$\exp_{\mathbf{X}}(\mathbf{y}) = \mathbf{X}^{\frac{1}{2}}\exp\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{y}\mathbf{X}^{-\frac{1}{2}}\right)\mathbf{X}^{\frac{1}{2}} \tag{A.30}$$

is a global diffeomorphism. Therefore, the logarithm is uniquely defined at all the points on the manifold

$$\log_{\mathbf{X}}(\mathbf{Y}) = \mathbf{X}^{\frac{1}{2}}\log\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}}\right)\mathbf{X}^{\frac{1}{2}}. \tag{A.31}$$

The exp and log are the ordinary matrix exponential and logarithm operators. Not to be confused, $\exp_{\mathbf{X}}$ and $\log_{\mathbf{X}}$ are manifold specific operators which are also point dependent, $\mathbf{X} \in Sym_d^+$. The tangent space of Sym_d^+ is the space of $d \times d$ symmetric matrices, and both the manifold and the tangent spaces are $m = d(d+1)/2$ dimensional.

For symmetric matrices, the ordinary matrix exponential and logarithm operators can be computed easily. Let $\mathbf{\Sigma} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ be the eigenvalue decomposition of a symmetric matrix. The exponential series is

$$\exp(\mathbf{\Sigma}) = \sum_{k=0}^{\infty}\frac{\mathbf{\Sigma}^k}{k!} = \mathbf{U}\exp(\mathbf{D})\mathbf{U}^T \tag{A.32}$$

where $\exp(\mathbf{D})$ is the diagonal matrix of the eigenvalue exponentials. Similarly, the logarithm is given by

$$\log(\mathbf{\Sigma}) = \sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}(\mathbf{\Sigma} - \mathbf{I})^k = \mathbf{U}\log(\mathbf{D})\mathbf{U}^T. \tag{A.33}$$

The exponential operator is always defined, whereas the logarithms only exist for symmetric matrices with positive eigenvalues, Sym_d^+.

The distance between two points $\mathbf{X}, \mathbf{Y} \in Sym_d^+$ is measured by substituting (A.31) into (A.29)

$$\begin{aligned} d(\mathbf{X}, \mathbf{Y}) &= [< \log_{\mathbf{X}}(\mathbf{Y}), \log_{\mathbf{X}}(\mathbf{Y}) >_{\mathbf{X}}]^{\frac{1}{2}} \\ &= \left[\operatorname{trace}\left(\log^2(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}})\right)\right]^{\frac{1}{2}}. \end{aligned} \tag{A.34}$$

We note that an equivalent form of the affine invariant distance metric was first given in [52], in terms of joint eigenvalues of $\mathbf{X}$ and $\mathbf{Y}$.

References

[1] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," in *Proc. European Conf. on Computer Vision,* Copehagen, Denmark, 2003, pp. 113–127.

[2] A. Aisenberg, "Coherent view of non-Hodgkin's lymphoma.," *J. Clin. Oncol.*, vol. 13, pp. 2656–2675, 1995.

[3] M. Alexa, "Linear combination of transformations," in *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, 2002, pp. 380–387.

[4] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3D point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, pp. 698–700, 1987.

[5] S. Avidan, "Ensemble tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, volume 2, 2005, pp. 494–501.

[6] S. Baker, R. Szeliski, and P. Anandan, "A layered approach to stereo reconstruction," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Santa Barbara, CA, 1998, pp. 434 – 441.

[7] E. Bayro-Corrochano and J. Ortegon-Aguilar, "Lie algebra template tracking," *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 2, pp. 56–59, 2004.

[8] E. Begelfor and M. Werman, "Affine invariance revisited," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 2, 2006, pp. 2087–2094.

[9] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 4, pp. 509–522, 2002.

[10] A. Berg, T. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, 2005, pp. 26–33.

[11] S. Birchfield and S. Rangarajan, "Spatiograms vs histograms for region-based tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, volume 2, 2005, pp. 1158–1163.

[12] M. Black and A. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *Intl. J. of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[13] S. Blostein and T. Huang, "Error analysis in stereo determination of 3D point position," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, pp. 752–765, 1987.

[14] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry.* Academic Press, second edition, 1986.

[15] N. Bouaynaya, W. Qu, and D. Schonfeld, "An online motion-based particle filter for head tracking applications," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* Philadelphia, volume 2, 2005, pp. 225–228.

[16] Y. Boykov and D. Huttenlocher, "Adaptive Bayesian recognition in tracking rigid objects," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC, volume 2, 2000, pp. 697–704.

[17] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 10, pp. 1042 – 1052, 1993.

[18] M. A. Carreira-Perpinan, "Gaussian mean-shift is an EM algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 5, pp. 767–776, 2007.

[19] F. Chabat, G. Yang, and D. Hansell, "Obstructive lung diseases: texture classification for differentiation at ct," *Radiology*, vol. 228, pp. 871–877, 2003.

[20] T.-J. Cham and J. M. Rehg, "A multiple hypothesis approach to figure tracking," in *In Proc. Perceptual User Interfaces*, 1998, pp. 19–24.

[21] J. Chan, P. Banks, M. Cleary, G. Delsol, C. De Wolf-Peeters, B. Falini, K. Gatter, T. Grogan, N. Harris, and P. Isaacson, "A revised European-American classification of lymphoid neoplasms proposed by the International Lymphoma study group - A summary version.," *Am. J. Clin. Pathol.*, vol. 103, pp. 543–560, 1995.

[22] H. Chen and P. Meer, "Robust fusion of uncertain information," *IEEE Trans. Systems, Man, Cybernetics-Part B*, vol. 35, pp. 578–586, 2005.

[23] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 790–799, 1995.

[24] S. Chirikjian and A. Kyatkin, *Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups.* CRC Press, 2001.

[25] R. Collins, "Mean shift blob tracking through scale space," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, volume 2, 2003, pp. 234–240.

[26] D. Comaniciu, "Variable bandwidth density-based fusion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, volume 1, 2003, pp. 56–66.

[27] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 603–619, 2002.

[28] D. Comaniciu, P. Meer, and D. Foran, "Image-guided decision support system for pathology," *Machine Vision and Applications*, vol. 11, pp. 213–224, 1999.

[29] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC, volume 1, 2000, pp. 142–149.

[30] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," in *Proc. European Conf. on Computer Vision,* Freiburg, Germany, 1998, pp. 484–498.

[31] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[32] R. Cotran, V. Kumar, T. Collins, and S. Robbins, *Pathologic Basis of Disease.* W.B. Saunders Company, fifth edition, 1994.

[33] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, 2000.

[34] O. Cula and K. Dana, "3D texture recognition using bidirectional feature histograms," *Intl. J. of Computer Vision*, vol. 59, no. 1, 2004.

[35] N. Dalal, *Finding people in images and videos.* PhD thesis, Institut National Polytechnique de Grenoble, July 2006.

[36] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, volume 1, 2005, pp. 886–893.

[37] B. C. Davis, P. T. Fletcher, E. Bullitt, and S. Joshi, "Population shape regression from random design data," in *Proc. 11th Intl. Conf. on Computer Vision,* Rio de Janeiro, Brazil, 2007, pp. 1–7.

[38] G. Dorkó and C. Schmid, "Selection of scale-invariant parts for object class recognition," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, 2003, pp. 634–640.

[39] T. Drummond and R. Cipolla, "Application of Lie algebras to visual servoing," *Intl. J. of Computer Vision*, vol. 37, pp. 21–41, 2000.

[40] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* Wiley, second edition, 2001.

[41] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *Proc. 7th Intl. Conf. on Computer Vision,* Kerkyra, Greece, September 1999, pp. 1033–1038.

[42] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3D rigid body transformations: A comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, pp. 272–290, 1997.

[43] A. Elgammal, R. Duraiswami, and L. S. Davis, "Efficient kernel density estimation using the efficient kernel density estimation using the color modeling and tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, pp. 1499–1504, 2003.

[44] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. European Conf. on Computer Vision,* Dublin, Ireland, volume 2, 2000, pp. 751–767.

[45] M. Fashing and C. Tomasi, "Mean shift is a bound optimization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, pp. 471–474, 2005.

[46] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *Intl. J. of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[47] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, 2003, pp. 264–271.

[48] P. Fletcher, C. Lu, and S. Joshi, "Statistics of shape via principal geodesic analysis on Lie groups," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, volume 1, 2003, pp. 95–101.

[49] P. T. Fletcher and S. Joshi, "Riemannian geometry for the statistical analysis of diffusion tensor data," *Signal Process.*, vol. 87, no. 2, pp. 250–262, 2007.

[50] D. Foran, D. Comaniciu, P. Meer, and L. Goodell, "Computer-assisted discrimination among lymphomas and leukemia using imunophenotyping, intelligent image repositories and telemicroscopy.," *IEEE Trans. on Infor. Tech. in Biomedicine*, vol. 4, pp. 265–273, 2000.

[51] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Intercommission Conf. on Fast Processing of Photogrammetric Data,* Interlaken, Switzerland, 1987, pp. 281–305.

[52] W. Förstner and B. Moonen, "A metric for covariance matrices," Technical report, Dept. of Geodesy and Geoinformatics, Stuttgart University, 1999.

[53] W. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *IEEE Intl. Workshop on Automatic Face and Gesture Recognition,* Zurich, Switzerland, 1995, pp. 296–301.

[54] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[55] N. Friedman and S. Russell, "Image segmentation in video sequences," in *Thirteenth Conf. on Uncertainty in Artificial Intelligence(UAI)*, 1997, pp. 175–181.

[56] K. Fukunaga and L. D. Hostetler, "Mean shift is a bound optimization," *IEEE Trans. Information Theory*, vol. 21, pp. 32–40, 1975.

[57] R. Galloway, "Texture analysis using gray level run lengths," *Comput. Graphic. Image Processing*, vol. 4, pp. 172–179, 1975.

[58] J. Garcia-Conde and F. Cabanillas, "Cell lymphoma: A lymphoproliferatie disorder associated with aberrant function of the cell cycle.," *Leukemia*, vol. 10, pp. 78–83, 1996.

[59] D. Gavrila and V. Philomin, "Real-time object detection for smart vehicles," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO, 1999, pp. 87–93.

[60] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis.* Chapman and Hall, second edition, 2003.

[61] B. Georgescu and P. Meer, "Point matching under large image deformations and illumination changes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, pp. 674–688, 2004.

[62] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, 2003, pp. 456–463.

[63] V. Govindu, “Lie-algebraic averaging for globally consistent motion estimation,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, volume 1, 2003, pp. 684–691.

[64] G. Hager and P. Belhumeur, “Efficient region tracking with parametric models of geometry and illumination,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 1025–1039, 1998.

[65] G. Hager, M. Dewan, and C. Stewart, “Multiple kernel tracking with SSD,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, volume 1, 2004, pp. 790–797.

[66] R. Haralick, “Statistical and structural approaches to texture,” *IEEE*, vol. 67, pp. 786–804, 1979.

[67] R. Haralick, K. Shanmugam, and I. Dinstein, “Textural features for image classification,” *IEEE Trans. Syst. Man Cybern.*, vol. 3, pp. 610–621, 1973.

[68] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proc. Alvey Vision Conf.*, 1988, pp. 147–151.

[69] M. Harville, G. Gordon, and J. Woodfill, “Foreground segmentation using adaptive mixture models in color and depth,” in *IEEE Workshop on Detection and Recognition of Events in Video*, 2001, pp. 3–11.

[70] K. Hassan, T. Tweed, and S. Miguet, “A multi-resolution approach for content-based image retrieval on the grid–application to breast cancer detection,” *Methods of information in medicine*, vol. 44, pp. 211–214, 2005.

[71] T. Hastie, R. Tibshirani, and J. Freidman, *The Elements of Statistical Learning.* Springer, 2001.

[72] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2nd edition, 1998.

[73] D. Heeger and J. Bergen, “Pyramid-based texture analysis/synthesis,” in *SIGGRAPH ’95: Proc. of the 22nd Annual Conf. on Computer Graphics and Interactive Techniques*, 1995, pp. 229–238.

[74] A. Hertzmann, C. Jacobs, N. Oliver, C. B., and D. Salesin, “Image analogies,” in *SIGGRAPH ’01: Proc. of the 28th Annual Conf. on Computer Graphics and Interactive Techniques*, 2001, pp. 327–340.

[75] B. Horn, H. Hilden, and S. Negahdaripour, “Closed-form solution of absolute orientation using orthonormal matrices,” *J. Opt. Soc. Am.*, vol. 5, pp. 1127–1135, 1988.

[76] C. Hsu and C. Lin, “A comparison of methods for multiclass support vector machines,” *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.

[77] Y. Huang, J. Chen, and S. W.C., “Diagnosis of hepatic tumors with texture analysis in nonenhanced computed tomography images,” *Academic radiology*, vol. 13, pp. 713–720, 2006.

[78] T. Ikeda and M. Hagiwara, “Content-based image retrieval system using neural networks,” *Intl. J. of Neural Systems*, vol. 10, pp. 417–424, 2000.

[79] S. Ioffe and D. A. Forsyth, “Probabilistic methods for finding people,” *Intl. J. of Computer Vision*, vol. 43, no. 1, pp. 45–68, 2001.

[80] M. Isard and I. Blake, “Condensation – conditional density propagation for visual tracking,” *Intl. J. of Computer Vision*, vol. 29, pp. 5–28, 1998.

[81] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, “Location of people in video images using adaptive fusion of color and edge information,” in *Proc. 15th Int’l Conf. on Pattern Recognition,* Barcelona, Spain, volume 4, 2000, pp. 627–630.

[82] M. Jaulent, C. Le Bozec, Y. Cao, E. Zapletal, and P. Degoulet, “A property concept frame representation for flexible image-content retrieval in histopathology databases,” in *AMIA Symp.*, 2000, pp. 379–383.

[83] K. Javed, O. Shafique and M. Shah, “A hierarchical approach to robust background subtraction using color and gradient information,” in *IEEE Workshop on Motion and Video Computing*, 2002, pp. 22–27.

[84] T. Joachims, *Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning.* MIT-Press, 1999.

[85] J. Jost, *Riemannian Geometry and Geometric Analysis.* Springer, fourth edition, 2005.

[86] F. Jurie and M. Dhome, “Hyperplane approximation for template matching,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 996–1000, 2002.

[87] T. Kadir and M. Brady, “Scale, saliency and image description,” *Intl. J. of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[88] K. Kanatani, *Group-Theoretical Methods in Image Understanding.* Springer-Verlag, 1990.

[89] E.-Y. Kang, I. Cohen, and G. Medioni, “Non-iterative approach to multiple 2D motion estimation,” in *Proc. 17th Int’l Conf. on Pattern Recognition,* Cambridge, UK, 2004, pp. 791–794.

[90] H. Karcher, “Riemannian center of mass and mollifier smoothing,” *Commun. Pure Appl. Math.*, vol. 30, pp. 509–541, 1977.

[91] K.-P. Karman and A. von Brandt, “Moving object recognition using an adaptive background memory,” in Capellini, editor, *Time-varying Image Processing and Moving Object Recognition*, volume 2, (Amsterdam, The Netherlands), Elsevier, 1990, pp. 297–307.

[92] D. Koller, J. Weber, and J. Malik, “Robust multiple car tracking with occlusion reasoning,” in *Proc. European Conf. on Computer Vision,* Stockholm, Sweden, 1994, pp. 189–196.

[93] F. Kuhn and C. Giardina, “Elliptic Fourier features of a closed contour,” *Computer Graphics Image Processing*, vol. 18, pp. 236–258, 1982.

[94] C. Le Bozec, E. Zapletal, M. Jaulent, H. D., and P. Degoulet, “Towards content-based image retrieval in a his-integrated pacs,” in *AMIA Symp.*, 2000, pp. 477–481.

[95] T. Lehmann, B. Wein, J. Dahmen, J. Bredno, F. Vogelsang, and M. Kohnen, "Content-based image retrieval in medical applications: A novel multistep approach," in *Proceedings of SPIE: Storage and Retrieval for Media Databases 2000*, volume 3972, 2000, pp. 312–320.

[96] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, volume 1, 2005, pp. 878–885.

[97] T. Leung and J. Malik, "Recognizing surfaces using three-dimensional textons," in *Proc. 7th Intl. Conf. on Computer Vision,* Kerkyra, Greece, 1999, pp. 1010–1017.

[98] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *Intl. J. of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[99] Y. Li, Y. Tsin, Y. Genc, and T. Kanade, "Object detection using 2D spatial ordering constraints," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, 2005, pp. 711 – 718.

[100] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[101] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *In Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

[102] J. Mao and A. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition*, vol. 25, pp. 173–188, 1992.

[103] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Random subwindows for robust image classification," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, volume 1, 2005, pp. 34–40.

[104] B. Matei and P. Meer, "Optimal rigid motion estimation and performance evaluation with bootstrap," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO, volume 1, 1999, pp. 339–345.

[105] I. Matthews and S. Baker, "Active appearance models revisited," *Intl. J. of Computer Vision*, vol. 60, pp. 135–164, 2004.

[106] M. Mavroforakis, H. Georgiou, N. Dimitropoulos, D. Cavouras, and S. Theodoridis, "Mammographic masses characterization based on localized texture and dataset fractal analysis using linear, neural and support vector machine classifiers," *Artif. Intell. Med.*, vol. 37, pp. 145–162, 2006.

[107] K. Mikolajczyk, B. Leibe, and B. Schiele, "Multiple object class detection with a generative model," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 1, 2006, pp. 26–36.

[108] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, 2001, pp. 525–531.

[109] K. Mikolajczyk, C. Schmid, and A. Zisserman, “Human detection based on a probabilistic assembly of robust part detectors,” in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, volume 1, 2004, pp. 69–81.

[110] E. Miller and C. Chef’dhotel, “Practical non-parametric density estimation on a transformation group for vision,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, volume 2, 2003, pp. 114–121.

[111] A. Mittal and N. Paragios, “Motion-based background subtraction using adaptive kernel density estimation,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, volume 2, 2004, pp. 302–309.

[112] M. Moakher, “A differential geometric approach to the geometric mean of symmetric positive-definite matrices,” *SIAM J. Matrix Anal. Appl.*, vol. 26, pp. 735–747, 2005.

[113] A. Mohan, C. Papageorgiou, and T. Poggio, “Example-based object detection in images by components,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 4, pp. 349–360, 2001.

[114] N. Mudigonda, R. Rangayyan, and J. Desautels, “Gradient and texture analysis for the classification of mammographic masses,” *IEEE Trans. Med. Imaging*, vol. 19, pp. 1032–1043, 2000.

[115] S. Munder and D. M. Gavrila, “An experimental study on pedestrian classification,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, pp. 1863–1868, 2006.

[116] B. Nielsen, F. Albregtsen, and H. Danielsen, “Prognostic value of adaptive textural features–the effect of standardizing nuclear first-order gray level statistics and mixing information from nuclei having different area,” *IEEE Trans. Medical Imaging*, vol. 23, pp. 73–84, 2004.

[117] B. Nielsen and H. Danielsen, “Prognostic value of adaptive textural features–the effect of standardizing nuclear first-order gray level statistics and mixing information from nuclei having different area,” *Cellular oncology*, vol. 28, pp. 85–95, 2006.

[118] N. Ohta and K. Kanatani, “Optimal estimation of three-dimensional rotation and reliability evaluation,” in *Proc. European Conf. on Computer Vision,* Freiburg, Germany, 1998, pp. 175–187.

[119] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer, “Weak hypotheses and boosting for generic object detection and recognition,” in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, 2004, pp. 71–84.

[120] A. Opelt, A. Pinz, and A. Zisserman, “Incremental learning of object detectors using a visual shape alphabet,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 1, 2006, pp. 3–10.

[121] P. Papageorgiou and T. Poggio, “A trainable system for object detection,” *Intl. J. of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[122] X. Pennec, P. Fillard, and N. Ayache, “A Riemannian framework for tensor computing,” *Intl. J. of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006.

[123] F. Porikli, “Integral histogram: A fast way to extract histograms in cartesian spaces,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, volume 1, 2005, pp. 829 – 836.

[124] F. Porikli and O. Tuzel, “Covariance tracking,” in *Video Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY.

[125] F. Porikli and O. Tuzel, “Multi-kernel object tracking,” in *Proc. of IEEE Int'l. Conference on Multimedia and Expo,* Amsterdam, Netherlands, 2005, pp. 1234–1237.

[126] F. Porikli and O. Tuzel, “Fast construction of covariance matrices for arbitrary size image windows,” in *IEEE Intl. Conf. on Image Processing*, 2006, pp. 1581–1584.

[127] F. Porikli, O. Tuzel, and P. Meer, “Covariance tracking using model update based on Lie algebra,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 1, 2006, pp. 728–735.

[128] L. Rajesh, S. Pattari, G. G., P. Dey, and R. Srinivasan, “Image morphometry of acute leukemias. comparison between lymphoid and myeloid subtypes.,” *Anal. Quant. Cytol. Histol.*, vol. 26, pp. 57–60, 2004.

[129] I. Ricketts, H. Banda-Gamboa, A. Cairns, and K. Hussein, “Automatic classification of cervical cells-using the frequency domain,” in *IEE Coll. on App. of Im. Proc. in Mass Health Screening*, volume 9, 1992, pp. 1–4.

[130] J. Rittscher, J. Kato, S. Joga, and A. Blake, “A probabilistic background model for tracking,” in *Proc. European Conf. on Computer Vision,* Dublin, Ireland, volume 2, 2000, pp. 336–350.

[131] R. Ronfard, C. Schmid, and B. Triggs, “Learning to parse pictures of people,” in *Proc. European Conf. on Computer Vision,* Copehagen, Denmark, volume 4, 2002, pp. 700–714.

[132] A. Rosenfeld and G. Vanderburg, “Coarse-fine template matching,” *IEEE Trans. Syst. Man. Cyb.*, vol. 7, pp. 104–107, 1977.

[133] W. Rossmann, *Lie Groups: An Introduction Through Linear Groups.* Oxford Press, 2002.

[134] H. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 22–38, 1998.

[135] C. Rozman and E. Montserrat, “Chronic lymphocytic leukemia.,” *The New England Journal of Medicine*, vol. 333, pp. 1052–1057, 1995.

[136] P. Sabzmeydani and G. Mori, “Detecting pedestrians by learning shapelet features,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.

[137] R. E. Schapire, “The boosting approach to machine learning: An overview,” in *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[138] C. Schmid, “Constructing models for content-based image retreival,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, HI, 2001, pp. 39–45.

[139] K. Schmid, N. Angerstein, S. Geleff, and A. Gschwendtner, "Quantitative nuclear texture features analysis confirms who classification 2004 for lung carcinomas," *Modern Pathology*, vol. 19, pp. 453–459, 2006.

[140] F. Schnorrenberg, C. Pattichis, C. Schizas, and K. Kyriacou, "Content-based retrieval of breast cancer biopsy slides," *Technology and Health Care*, vol. 8, pp. 291 – 297, 200.

[141] J. Selig, *Geometric Methods in Robotics.* Springer-Verlag, 1996.

[142] H. Sheshadri and A. Kandaswamy, "Experimental investigation on breast tissue classification based on statistical feature extraction of mammograms," *Comput. Med. Imaging Graph.*, vol. 31, pp. 46–48, 2007.

[143] H. Shum and R. Szeliski, "Construction of panoramic image mosaics with global and local alignment," *Intl. J. of Computer Vision*, vol. 16, pp. 63–84, 2000.

[144] P. Simard, L. Bottou, P. Haffner, and Y. LeCun, "Boxlets: A fast convolution algorithm for signal processing and neural networks," in *Proc. Advances in Neural Inf. Proc. Sys. II*, 1998, pp. 571–577.

[145] L. Sirovitch and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America*, vol. 2, pp. 519–524, 1987.

[146] C. Stauffer and E. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO, volume 2, 1999, pp. 246–252.

[147] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Buhmann, "Topology free hidden Markov models: Application to background modeling," in *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, 2001, pp. 294–301.

[148] R. Subbarao and P. Meer, "Nonlinear mean shift for clustering over analytic manifolds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 1, 2006, pp. 1168–1175.

[149] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 39–51, 1998.

[150] A. Torralba, K. Murphy, and W. Freeman, "Sharing features: Efficient boosting procedures for multiclass object detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, 2004, pp. 762–769.

[151] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. 7th Intl. Conf. on Computer Vision,* Kerkyra, Greece, 1999, pp. 255–261.

[152] M. Turk and A. Pentland, "Face recognition using eigenfaces," 1991, pp. 586–591.

[153] O. Tuzel, F. Porikli, and P. Meer, "A bayesian approach to background modeling and low frame rate tracking," in *IEEE Int. Workshop on Machine Vision for Intelligent Vehicles,* San Diego, CA, 2005.

[154] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Proc. European Conf. on Computer Vision,* Graz, Austria, volume 2, 2006, pp. 589–600.

[155] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on Riemannian manifolds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.

[156] O. Tuzel, F. Porikli, and P. Meer, "Learning on Lie groups for invariant detection and tracking," in *To appear in Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Anchorage, AK, 2008.

[157] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on Riemannian manifolds," *To appear in IEEE Trans. Pattern Anal. Machine Intell.*, 2008.

[158] O. Tuzel, R. Subbarao, and P. Meer, "Simultaneous multiple 3D motion estimation via mode finding on Lie groups," in *Proc. 10th Intl. Conf. on Computer Vision,* Beijing, China, volume 1, 2005, pp. 18–25.

[159] O. Tuzel, L. Yang, P. Meer, and D. Foran, "Classification of hematologic malignancies using texton signatures," *Pattern Analysis and Applications*, vol. 10, pp. 277–290, 2007.

[160] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 376–380, 1991.

[161] G. Vadlamudi, "Leukemic phase of mantle cell lymphoma: Two case reports and review of the literature.," *Arch. Pathol. Lab. Med.*, vol. 120, pp. 35–40, 1996.

[162] M. Varma and A. Zisserman, "Statistical approaches to material classification," in *Proc. European Conf. on Computer Vision,* Copehagen, Denmark, 2002.

[163] M. Varma and A. Zisserman, "Texture classification: Are filter banks necessary?," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, volume 2, June 2003, pp. 691–698.

[164] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *Intl. J. of Computer Vision*, vol. 68, pp. 7–25, 2005.

[165] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, HI, volume 1, 2001, pp. 511–518.

[166] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 1, 2003, pp. 734–741.

[167] R. Walker and P. Jackway, "Statistical geometric features-extensions for cytological texture analysis," in *Proc. 13th Intl. Conf. on Pattern Recognition*, volume 2, 1996, pp. 790–794.

[168] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, "Image and video segmentation by anisotropic kernel mean shift," in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, volume 2, 2004, pp. 238–249.

[169] M. Weber, M. Welling, and P. Perona, "Unsupervised learning of models for recognition," in *Proc. European Conf. on Computer Vision,* Dublin, Ireland, 2000, pp. 18–32.

[170] Y. Weiss, "Smoothness in layers: Motion segmentation using nonparametric mixture estimation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Juan, Puerto Rico, 1997, pp. 520–527.

[171] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian learning for efficient visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, pp. 1292–1304, 2005.

[172] C. Wren, A. Azarbayejani, T. Darell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, 1997.

[173] G. Wyszecki and W. Stiles, *Concepts and Methods, Quantitative Data and Formulae.* Wiley, New York, 1982.

[174] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Trans. on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[175] C. Yang, R. Duraiswami, D. DeMenthon, and L. Davis, "Mean-shift analysis using quasi-Newton methods," in *IEEE Intl. Conf. on Image Processing*, volume 2, 2003, pp. 447–450.

[176] L. Yang, P. Meer, and D. Foran, "Unsupervised segmentation based on robust estimation and color active contour models," *IEEE Trans. on Information Technology in Biomedicine*, vol. 9, pp. 475–486, 2005.

[177] L. Zelnik-Manor and M. Irani, "Multiframe estimation of planar motion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 1105–1115, 2000.

[178] T. Zhao and R. Nevatia, "Tracking multiple humans in crowded environment," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, volume II, 2004, pp. 406 – 413.

[179] S. Zhou, R. Chellappa, and B. Moghaddam, "Visual tracking and recognition using appearance-based modeling in particle filters," *IEEE Trans. on Image Processing*, vol. 13, pp. 1491–1506, 2004.

[180] S. Zhou, J. Zhou, and D. Comaniciu, "A boosting regression approach to medical anatomy detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.

[181] Q. Zhu, S. Avidan, M. C. Yeh, and K. T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, volume 2, 2006, pp. 1491 – 1498.

Vita

Cuneyt Oncel Tuzel

Education

2003-2008 Ph.D., Computer Science, Rutgers University

1999-2003 Master of Science, Computer Engineering, Middle East Technical University, Turkey

1995-1999 Bachelor of Science, Computer Engineering, Middle East Technical University, Turkey

Publications

Journal

- **O. Tuzel**, F. Porikli, and P. Meer, "Pedestrian detection via classification on Riemannian manifolds", to appear in *IEEE Trans. Pattern Anal. Machine Intelligence.*
- **O. Tuzel**, L. Yang, P. Meer, and D.J. Foran, "Classification of hematologic malignancies using texton signatures", in *Pattern Analysis and Applications*, Vol. 10, pp. 277-290, 2007.
- O.F. Ozer, O. Ozun, **O. Tuzel**, V. Atalay, and E. Cetin, "Vision-based single-stroke character recognition for wearable computing", in *IEEE Intelligent Systems*, Vol.16, Issue 3, pp. 33-37, 2001.

Conference

- L. Yang, **O. Tuzel**, P. Meer, and D.J. Foran, "Automatic Image Analysis of Histopathology Specimens Using Concave Vertex Graph", to appear in *Proc. 11th Intl. Conf. on Medical Image Computing and Computer Assisted Intervention,* New York City, USA, 2008.
- **O. Tuzel**, F. Porikli, and P. Meer, "Learning on Lie groups for invariant detection and tracking", to appear in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Anchorage, Alaska, 2008.

- **O. Tuzel**, F. Porikli, and P. Meer, "Human detection via classification on Riemannian manifolds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007, **Best paper prize runner-up**.

- F. Porikli, and **O. Tuzel**, "Fast construction of covariance matrices for arbitrary size image windows", in *Proc. IEEE Int. Conf. on Image Processing*, pp.1581-1584, 2006.

- F. Porikli, **O. Tuzel**, and P. Meer, "Covariance tracking using model update based on Lie algebra," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, Vol. 1, pp. 728-735, 2006.

- F. Porikli, and **O. Tuzel**, "Covariance tracker," in *Video Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, 2006.

- **O. Tuzel**, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Proc. 9th European Conf. on Computer Vision,* Graz, Austria, Vol. 2, pp. 589–600, 2006.

- F. Porikli, and **O. Tuzel**, "Bayesian background modeling for foreground detection", in *Proc. 3rd ACM Int. Workshop on Video surveillance & sensor networks,* Singapore, pp. 55-58, 2005.

- **O. Tuzel**, R. Subbarao, and P. Meer, "Simultaneous multiple 3D motion estimation via mode finding on Lie groups", in *Proc. 10th Intl. Conf. on Computer Vision,* Beijing, China, Vol. 1, 2005, pp. 18-25.

- F. Porikli, and **O. Tuzel**, "Multi-kernel object tracking", in *Proc. IEEE Int. Conf. on Multimedia and Expo,* Amsterdam, Netherlands, pp. 1234-1237, 2005.

- **O. Tuzel**, F. Porikli, and P. Meer, "A Bayesian approach to background modeling," in *IEEE Int. Workshop on Machine Vision for Intelligent Vehicles,* San Diego, CA, 2005.

- F. Porikli, and **O. Tuzel**, "Human body tracking by adaptive background models and mean-shift analysis," in *IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.