SELECTION-BASED DICTIONARY LEARNING FOR SPARSE REPRESENTATION IN VISUAL TRACKING

BY BAIYANG LIU

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
Casimir A. Kulikowski

and approved by

New Brunswick, New Jersey
October, 2012
ABSTRACT OF THE DISSERTATION

Selection-based Dictionary Learning for Sparse Representation in Visual Tracking

by Baiyang Liu

Dissertation Director: Casimir A. Kulikowski

This dissertation describes a novel selection-based dictionary learning method with a sparse representation to tackle the object tracking problem in computer vision. The sparse representation has been widely used in many applications including visual tracking, compressive sensing, image de-noising and image classification, and learning a good dictionary for the sparse representation is critical for obtaining high performance. The most popular existing dictionary learning algorithms are generalized from K-means, which compute the dictionary columns to minimize the overall target reconstruction error iteratively. For better discriminative capability to differentiate target-object (positive) from background (negative) data, a class of dictionary algorithms has been developed to learn the dictionary from both the positive and the negative data. However, these methods do not work well for visual tracking in a dynamic environment in which the background can change considerably between frames in a non-linear way. The background cannot be modeled statically with the usual linear models. In this dissertation, I report on the development of a selection-based dictionary learning algorithm (K-Selection) that constructs the dictionary by choosing its columns from the training data. Each column is the most representative basis for the whole dataset, which also has a clear physical meaning. With locality-constraints, the subspace represented by the learned dictionary is not restricted to the training data alone, and is also less sensitive to outliers. The sparse representation based on
this dictionary learning method supports a more robust tracker trained on the target-object data alone. This is because the learned dictionary has more discriminative power and can better distinguish the object from the background clutter. By extending the dictionary with encoded spatial information, I present a new tracking algorithm which is robust to dynamic appearance changes and occlusions. The performance of the proposed algorithms have been validated for several challenging visual tracking applications through a series of comparative experiments.
**List of Figures**

1.1. Examples of 2D visual tracking .......................... 2
1.2. An example of 3D object tracking ........................ 2
1.3. Example Applications of Visual Tracking ................ 4
1.4. Challenges of visual tracking in dynamic environment .... 5
1.5. Challenges of visual tracking in dynamic environments II 6
1.6. General modules of visual tracking ........................ 7
1.7. Approximate LLC ....................................... 12
2.1. Dynamic Group Sparsity in Tracking ...................... 18
2.2. Comparative tracking result of sequence 1 ............... 25
2.3. Comparative tracking result of sequence 2 ............... 26
2.4. Comparative tracking result of sequence 3 ............... 27
2.5. Comparative tracking results of sequence 4 ............... 28
2.6. Comparative tracking result of sequence 5 ............... 29
2.7. Quantitative tracking accuracy of experimented sequences 30
3.1. Illustration of initial dictionary selection ............... 34
3.2. K-Selection Iterative Updates ............................ 35
3.3. Performance of the learned dictionaries using K-Selection and K-SVD 37
4.1. Tracking Framework of The Sparse Tracking using Local Appearance Model 40
4.2. Example probability map (right) of the relative target location for a single dictionary patch (labeled by the green rectangle) in the left image. The template patches labeled by red rectangles have non-zero coefficients for this dictionary patch and therefore contribute to the probability map. 44
4.3. Analysis of the tracking performance with different sizes of patches 48
4.4. Comparative results on benchmark sequences ......................... 49
4.5. Tracking results of selected frames from the David sequence ............... 55
4.6. Tracking result of selected frames from the Girl sequence .................... 56
4.7. Tracking result of selected frames from the Car sequence ..................... 57
4.8. Tracking result of selected frames from the Faceoc2 sequence ................. 58
4.9. Tracking result of selected frames from the Lemming sequence ................ 59
4.10. Tracking result of selected frames from the Box sequence ................... 60
4.11. Comparative results on the PROST sequences .............................. 61
4.12. Comparative tracking result on the Board sequence and the Liquor sequence .. 62
4.13. Tracking results of celebrity face data ...................................... 63
4.14. Tracking experiments on sequences with more motion dynamics ............ 63
4.15. Result comparison of static voting map and online updated voting map .... 64
5.1. The sample fluoroscopic image. The tumor contour is labeled in the right figure. ... 66
5.2. The computation of the motion prior of the tumor using nonlinear registration of the lung between consecutive frames. ......................... 68
5.3. The comparative tracking results of four sequences (a-d) with proposed method (top row), online collaborative tracking (middle row) and optical flow (bottom row). .................. 73
Acknowledgements

This dissertation was possible only through the support and help of my families, colleagues and friends.

First, I want to thank my advisor Professor Casimir Kulikowski for his guidance and support in my Ph.D. studies and research. I have learned tremendously from him about both academic research and the attention to personal and social factors that are so critical for carrying it out successfully. All my publications would have been impossible without his detailed revision and guidance. I still remember how hard it was to find the right research direction for a new student who had just graduated from college. Professor Kulikowski kept encouraging me and helping me explore different topics, including imaging processing, clustering, bioinformatics, etc. All this knowledge has been extremely valuable for me and has contributed to this dissertation.

I would like to thank Dr. Leiguang Gong for his great support since the days of my undergraduate studies. Without his generous help and guidance throughout my entire Ph.D. studies, it would have also been impossible for me to reach the present milestone. Dr. Leiguang Gong taught me how to do research, how to select a topic, how to communicate with colleagues and so on. I have to sincerely thank him for so much that I have learned from him.

I also would like to thank my colleagues Prof. Lin Yang, now at the University of Kentucky, and Prof. Junzhou Huang at the University of Texas at Arlington for our collaboration when they were here at Rutgers and UMDNJ. All the insightful and deep discussions with them inspired the key flow of ideas presented in this dissertation.

I also want to thank Professor Ahmed Elgammal and Professor Vladimir Pavlovic for their valuable advice in helping me to improve the quality of my research and this dissertation. I took the machine learning class given by Professor Elgammal which built the foundation of my later research in computer vision.
I also want to thank Professor Peter Meer from whom I learned critical insights and approaches to computer vision in two courses I took with him. He helped me learn not only the foundational knowledge for computer vision, but also the ways of doing research in this field.

The six years at Rutgers doing the research described here have been quite amazing, with much happiness but also many challenges. I have to thank many friends for sharing in the former and helping overcome the latter. I could not have kept walking along this long road towards a PhD, and overcome so many difficulties without all the encouragement and support from my friends. I would like to especially thank Shaoting Zhang, Xinyi Cui, Fuyong Xing and Xin Qi for our valuable discussion on research, studies, and related experiences. I also want to express my appreciation to Hong Yang, Jun Tan, Dong Dai and so many other friends for sharing such unforgettable experiences and memories.

Finally, I want to thank my parents and my wife for their tremendous support. In the final year of getting my doctoral work to converge, I will always remember the times when I had to stay long and late hours in the lab while my wife prepared my meals at home, always supporting and encouraging me to improve my work and successfully pursue my research career.
Dedication

To my parents, my wife and our daughter, princess Fiona!
# Table of Contents

Abstract ................................................................. ii

List of Figures ............................................................ iv
Acknowledgements ........................................................ vi
Dedication ................................................................. viii

1. Introduction .......................................................... 1
   1.1. Motivation ...................................................... 1
       1.1.1. Visual Tracking ........................................ 1
       1.1.2. Example Applications ................................. 2
       1.1.3. Challenges ............................................. 3
   1.2. Visual Tracking Framework ................................... 7
   1.3. Sparse Coding for Object Representation ..................... 8
       1.3.1. Sparse Coding ........................................... 9
       1.3.2. Orthogonal Matching Pursuit ....................... 10
       1.3.3. Dynamic Group Sparsity ............................. 10
       1.3.4. Locality-constrained Linear Coding ............... 11
   1.4. Sparse Representation in Tracking .......................... 12
   1.5. Sparse Dictionary Learning .................................. 13
   1.6. Contributions of the Dissertation .......................... 14
   1.7. dissertation Overview ...................................... 15

2. Two Stage Sparse Learning in Visual Tracking .................. 16
   2.1. Introduction .................................................. 16
   2.2. Related Work ............................................... 17
2.3. Bayesian tracking framework ........................................... 19
2.4. Two-stage sparse representation ..................................... 21
   2.4.1. Feature selection .................................................. 22
   2.4.2. Sparse Reconstruction .......................................... 23
2.5. Experimental Results .................................................. 24
   2.5.1. Visual Evaluation of Comparative Experimental Results .... 26
   2.5.2. Quantitative Evaluation of Comparative Experimental Results 28
2.6. Conclusion .............................................................. 29

3. Selection-based Dictionary Learning ................................. 31
   3.1. Motivation ............................................................ 31
   3.2. Selection-based Dictionary Learning ............................... 32
      3.2.1. Objective function ............................................. 32
      3.2.2. K-Selection .................................................... 32
      3.2.3. Initial Dictionary Selection .................................. 33
      3.2.4. Iterative Optimization ....................................... 34
   3.3. Applications Exploration .......................................... 35
      3.3.1. Target Modeling in Visual Tracking ......................... 35
      Reconstruction vs. Discriminative Power ......................... 35

4. Robust tracking using a local sparse appearance model ............ 38
   4.1. Introduction .......................................................... 38
   4.2. Tracking Framework ............................................... 39
   4.3. Target Modeling ..................................................... 39
      4.3.1. Sparse Coding Histogram .................................... 39
   4.4. Target Localization ................................................ 41
      4.4.1. Sparse Constraint Regularized Mean-shift .................. 41
      4.4.2. Voting in a Sparse Representation ......................... 43
4.5. Experimental Result ......................................................... 47
  4.5.1. Parameter Analysis ..................................................... 48
  4.5.2. Comparative Tracking Results ...................................... 48
4.6. Discussion ...................................................................... 52

5. Motion Registration Guided Online Collaborative Trackers ............... 65
  5.1. Introduction ................................................................... 65
  5.2. Online Contour Tracking Framework .................................. 66
  5.3. Tumor Contour Prediction Using Lung Motion Registration .......... 67
  5.4. Online Collaborative Tracking .......................................... 69
    5.4.1. Asymmetric Online Boosting ....................................... 70
    5.4.2. Dynamic Appearance Model and Subspace Learning .......... 71
  5.5. Experimental Results ..................................................... 72
  5.6. Conclusion ..................................................................... 74

6. Conclusions and Future Directions ............................................... 75

References ............................................................................ 76
Chapter 1

Introduction

1.1 Motivation

1.1.1 Visual Tracking

Computer vision has been an active research field spanning computer science and engineering, and become an ubiquitous source of applications for the past few decades. By modeling the human perception, computer vision algorithms can perform similar tasks to those performed by human beings, and can even perform better or more efficiently for some types of tasks than humans. These include massive video data analysis, hazardous environment vision with robotics, etc.. Among the many challenging problems in Computer Vision, object motion tracking is one that requires much ingenuity to solve. Visual tracking needs to estimate the spatial state of a specified target from digital video recordings using visual cues from the target object. Even with complicated dynamic environments humans can handle this task, yet it remains challenging for automatic analysis carried out by computer algorithms. In this dissertation, I report on of the design, implementation, and results of modeling the appearance of a target, in order to make inferences about its spatial states, and so as to construct a robust tracker using a sparse and efficient representation of the object.

Given a video sequence, visual object tracking can be defined as estimating the spatial states of a specified target and recognizing the type of objects in each frame of a video sequence. One definition of spatial states would be the location of the target’s center in a 2D image frame, or an object’s scale or rotation angle as shown in Figure 1.1. Using more than one camera or additional specification of the 3D environment, lot of research work has carried out to estimate the 3D location and pose of a given target such as the tracking example in Figure 1.2.

In this dissertation, the focus is on the problem of tracking objects in 2D video image frames
with appearance projected from the 3D world. To be more specific, given a set of consecutive video image frames $I_{t,t=1..T}$ and an object $O$, object tracking can be formulated as inferring the spatial state $X_{o,t}$ of the object in the $t$th frame.

Even thought lot of research work has been carried out on this problem during the past decades, optimal solutions have yet to be explored.

### 1.1.2 Example Applications

Object tracking itself plays a very important role in many other interesting computer vision applications that rely on the accurate estimation of the target location, motion or trajectory. Several example applications that are directly related to object tracking are presented in Figure 1.3.
In security surveillance, detecting the moving object and tracking its trajectory are necessary for adjusting the angle of the security camera, so that the camera can follow the target. Further analysis of the tracked result can be used for abnormality detection, social scene analysis and group action recognition. Figure 1.3(a) is an example of indoor surveillance.

Figure 1.3(b) shows an example application of vehicle velocity estimation for traffic surveillance. With the help of tracking technology, it is feasible to estimate the location, velocity and trajectory of all vehicles. This information can be further analyzed for traffic monitoring without manual observation.

Another major field of tracking applications is medical image analysis. Computer vision algorithms can analyze massive medical data and provide useful information to a physician so they can reach a more accurate and objective diagnosis. Figure 1.3(c) presents an example application of tracking in live surgery provided by Siemens. The guide-wire and catheter are tracked online and visualized for the doctor during surgery.

In Figure 1.3(d), the tracking algorithm was told to locate the contour of a lung tumor during respiration cycles. Accurate estimation of the tumor location and shape boundary can help the radiologist to determine the dose of radiation. It is very important for the patient to destroy tumor cells in a way that causes minimum damage to nearby healthy cells. However, distinguishing the ill-defined boundary of a lung tumor in some frames may prove most difficult even for a human expert. With the help of tracking algorithms, it is possible for computer algorithms to infer the boundary by systematically capturing more detailed information than cannot be noticed by human beings. More discussion of this application follows in Chapter 5.

1.1.3 Challenges

In this dissertation, I focus on the problem of tracking a single object in a 2D video frame with a dynamically changing environment. Both the target and the background are moving and dynamic. However, even this simplified tracking problem is challenging as shown in Figure 1.4 and Figure 1.5

- In dynamic environments, both the target and camera move around. The target may experience illumination changes, rotation, and also scaling due to the varying distance
Figure 1.3: 1.3(a). Application of action and scene recognition [3]; 1.3(b). Application of traffic monitoring [20]; 1.3(c). Application of guide-wire tracking for live surgery [50]; 1.3(d). Tumor contour motion tracking [29]
Figure 1.4: Challenges of visual tracking in dynamic environment
Figure 1.5: Challenges of visual tracking in dynamic environment II

(a) Heavy illumination change

(b) Heavy occlusions
between the target and the camera:

- Tracking the motion of a target by the human eye is often easy, as we can easily (re)construct the target and environment in 3D space. However, the main step in digital video acquisition is projecting the 3D world to a 2D camera world. The information loss from 3D to 2D makes the estimation of the pose variation challenging;

- Partial or full object occlusions reduce the visible areas of the target. How to accurately locate the target under occlusion is difficult. But it is very important for achieving robustness in tracking. Mistaken learning of occlusion is discussed separately, and is usually a cause of drifting;

- The difficulty of tracking a specified target also increases when there are other similar objects from the same category that appear in the video frames. How to construct a tracker with high discriminative ability is very important for the robustness of the process.

1.2 Visual Tracking Framework

There are two major subtasks which can be represented as sub-modules in visual tracking. These are shown in 1.6: target modeling and target localization. The object tracking starts from specifying the target at the beginning. The target can be specified by the user or can be detected by a pre-trained object detector, such as a face detector, a pedestrian detector, and so on. The target model can then be learned so as to represent the specified target using its appearance features, including color, intensity, shape or other advanced features. Then the localization module is required to locate the target in a given frame, or classify the sample candidates generated from the current frame to be either target or background.

How to model the appearance of the target is very critical for robust tracking. Generative
and discriminative methods are the two major approaches used in current tracking techniques. The generative models formulate the tracking problem as searching for the regions with the highest likelihood [6, 11, 34, 39, 41, 53, 58, 33], either using a single appearance model or multiple appearance models [56]. Discriminative methods formulate the tracking as a classification problem [4, 5, 8, 21]. The trained classifier is used to discriminate the target from the background and can be updated online. Grabner et. al. [17] proposed to update the selected features incrementally using the tracking result of the current frame, but this can lead to potential drifting due to the accumulation of errors.

In order to handle the drifting problem, semi-online boosting [18] has been proposed to incrementally update the classifier using both unlabeled and labeled data. The Multiple Instance Learning boosting method (MIL) [5] puts all samples into bags and labels them. The drifting problem is handled in this method since the true target included in the positive bag is learned implicitly. Recently, it has been shown that an appropriate combination of complementary tracking algorithms can alleviate drifting problems [25, 55, 44, 42].

In [35, 28], a sparse representation was introduced for tracking and modeling the dynamic appearance of the target. The occlusion was modeled as a sparse noise component. All these methods model the target as a single entity, and therefore cannot handle partial occlusion very well. Fragment-based tracking in [1] coupled with a voting map can accurately track a partially occluded target. However, this method tracks each target patch with a static template, which limits its expressive power. It may fail in a dynamic environment which exhibits appearance changes or pose variations.

1.3 Sparse Coding for Object Representation

Sparse representations have recently attracted much interest in computer vision and machine learning. Recent publications have shown that the sparse representation can be used in many applications including image denoising [15], inpainting [43], classification [40], segmentation [31], visual tracking [27] and face recognition [57, 24], etc. In this section, related algorithms for sparse coding are discussed.
1.3.1 Sparse Coding

With the sparse assumption, the given signal $x$ can be represented as the linear combination of a few basis vectors in the collected library $\Phi$ with component $\epsilon$ representing the noise

$$x = \Phi \alpha + \epsilon. \quad (1.1)$$

The general sparse problem can be formulated as finding the sparse coefficients $\alpha$ that can achieve some maximum tolerable reconstruction error.

$$\alpha_0 = \arg\min_\alpha ||\alpha||_0 \text{ subject to } ||x - \Phi \alpha|| < \epsilon \quad (1.2)$$

where $||.||_0$ denotes the zero norm which represents the number of nonzero components and $\epsilon$ is the level of the reconstruction error. However, it is well known that the $l_0$ optimization problem is NP-hard and there is no efficient algorithm to find the global optimum solution other than exhaustive search.

One class of algorithms tries to seek the sparsest solution by performing basis pursuit (BP) based $l_1$ minimization as

$$\alpha_* = \arg\min_\alpha ||x - \Phi \alpha|| + \tau ||\alpha||_1 \quad (1.3)$$

using linear programming instead of optimizing the $l_0$ minimization in Equation 1.2 [14].

Another well known class of sparse optimization algorithms involves iterative greedy pursuit. The earliest algorithms included matching pursuit [32] and orthogonal matching pursuit [46]. Subspace pursuit [12] and compressive sampling matching pursuit [38] were proposed to achieve theoretical recovery guarantees similar to those of the BP while reducing computational complexity. Considering the group structure properties of the sparse coefficients, Dynamic Group Sparse (DGS) learning has been proposed to solve the sparse coding problem in [22] and been used in tracking [28]. In [48], a locality-constrained linear coding algorithm was proposed to emphasize the locality over sparsity which exactly fits tracking applications with appearance similarity.
1.3.2 Orthogonal Matching Pursuit

An efficient algorithm for sparse coding problem has been proposed for signal recovery in [46]. It is assumed that the columns of $\Phi$ has been normalized so that $|\Phi_i|_2 = 1$ for column $i = 1, 2, ..., K$. Let $S \subseteq 1, 2, 3, ..., K$ as the index subset of $\Phi$, the problem can be formulated as finding a subset $S$ with corresponding coefficient $\alpha_S \neq 0$.

$$\argmin ||x - \Phi \alpha||^2_2, \text{ subject to } ||\alpha||_0 \leq l$$

The OMP algorithm finds $S$ and the corresponding $\alpha_S$ iteratively as stated below:

- Step 1: Initialize the residual $r_0 = x$ and the selected subset as $\emptyset$, $i = 1$.
- Step 2: Find the columns $\Phi_{t^*}$ that have the largest correlation with the current residual and add $t^*$ to the selected subset $S_i = S_{i-1} \cup t^*$

$$t^* = \argmax_i |\Phi'_i r_{i-1}|$$

- Step 3: Update $r_i = (I - P_i)x$, where $P_i$ is the projection onto the linear subspace spanned by the selected subset $\Phi_{S_i}$.
- Step 4: Terminate the algorithm if the stopping condition is reached. Otherwise, repeat from step 2 with $i = i + 1$.

1.3.3 Dynamic Group Sparsity

It was observed that the non-zero coefficients in the solution $\alpha$ are not randomly distributed in some practical applications by [22]. So the Dynamic Group Sparsity (DGS) algorithm was proposed to solve the sparse coding problem which tends to cluster the non-zero components into groups.

The DGS algorithm is extended from the Subspace Pursuit (SP) [12] algorithm described as following:

- Step 1: Prune the residue estimation;
• Step 2: Merge the support sets;

• Step 3: Solve the least square problem with the current support sets;

• Step 4: Prune the signal estimation;

• Step 5: Update the support set and residual estimation;

The advantage of using DGS in tracking will be discussed in section 2.2.

1.3.4 Locality-constrained Linear Coding

Besides the group property of the non-zero components, locality is another preferred feature of the sparse solution for some applications. An efficient algorithm called Locality-constrained Linear Coding (LLC) has been proposed by [48]. In LLC, the objective function is changed to:

$$\min_\alpha ||x - \sum_{i=1}^{K} \Phi_i \alpha_i||^2 + \lambda ||d \odot \alpha||^2,$$

$$s.t. 1^T \alpha = 1$$ (1.6)

where $\odot$ denotes element-wise multiplication and $d$ is exponential to the Euclidean distance vector between $y$ and all basis vectors in $\Phi$. Constraint $1^T \alpha = 1$ ensures shift-invariance. The solution $\alpha$ is not $l_0$ norm sparse, but has only a few significant components. The second term in Eqn. (1.6) penalizes the distance from $x$ to the basis $\Phi$. LLC selects a set of local basis vectors for $x$ to form a local coordinate system. Thus, a faster but approximate LLC can be derived by solving a smaller linear system $B$ containing the $k$ nearest neighbors of $x$ as shown in Figure 1.7.

$$\min_{\hat{\alpha}} ||x - B \hat{\alpha}||^2,$$

$$s.t. 1^T \hat{\alpha} = 1$$ (1.7)

Since $B$ is a subset of $\Phi$ and $\hat{\alpha}$ in Eqn. (1.7) is a dense solution, $\hat{\alpha}$ can be considered as the non-zero coefficients in the original $\alpha^*$ of LLC corresponding to the components in $B$ with the
coefficients of all other components set to 0. In terms of tracking, LLC is used to compute the representation with the local template (basis) in \( \Phi \) that is similar to the candidate sample \( x \).

### 1.4 Sparse Representation in Tracking

In visual tracking, the sparse representation can be used to model the target appearance and measure the likelihood of a given candidate to be the target. Given \( N \) training samples of the target appearance \( x_i, \ i = 1, 2, ..., N \), a dictionary of the target appearance can be learned from the given samples, which can be considered as the appearance model of the target. Then the likelihood of a given candidate for being the target can be measured by the minimum reconstruction error from the learned appearance model. It is valid to assume that the target samples will have a smaller reconstruction error than the background samples. Thus, the likelihood of a target sample with smaller reconstruction error should be larger than for background samples. Assume that similar target appearances fall on a linear subspace and then any target samples can be represented as a linear combination of some other target samples. It has also be observed that only a small number of templates are necessary to construct a true target sample with small reconstruction error. Let \( \Phi \) be the learned dictionary from target training data. The representation can be formulated as

\[
x = \Phi \alpha + f, \ \text{subject to} |\alpha|_0 < k_1 \ \text{and} |f|_0 < k_2
\]  

In visual tracking, the first component is the sparse representation of candidate \( x \) with \( f \).
as the sparse occlusion or noise component. \( k_1 \) and \( k_2 \) constrain the sparsity of the linear coefficients \( \alpha \) and the occlusion component \( f \) respectively.

This sparse representation problem can be reformulated as a \( l_1 \) minimization problem and solved as in [35]. The representation coefficient \( \alpha \) can be computed by optimizing the \( l_1 \) regularized least square problem, which typically provides a sparse solution [49]:

\[
\alpha^* = \text{argmin}_\alpha ||x - \Phi \alpha - f||^2 + \lambda ||\alpha'||_1
\]

where \( \alpha' = (\alpha^T, f^T)^T \) and the parameter \( \lambda \) controls the sparsity of both coefficient vector and noise.

In this dissertation, we focus on solving the sparse representation using the iterative methods discussed in Section 1.3.1, which are more efficient and intuitive. With the solved sparse coefficients, the likelihood \( P(x|\Phi) \) of candidate \( x \) being generated from the target appearance model can be calculated as:

\[
p(x|\Phi) = Ce^{-\epsilon^2/\sigma^2}, \quad \epsilon = ||x - \Phi \alpha^*||_2.
\]

### 1.5 Sparse Dictionary Learning

A dictionary \( \Phi \) is assumed to be given in the above description. A good dictionary is critical for achieving high performance across different applications. However, it is not trivial to learn a dictionary for a sparse representation. In this section, the formulation of dictionary learning and related literature are reviewed.

For a given training data set with \( N \) signal \( x_i, i = 1, 2, ..., N \), the dictionary learning problem can be formulated as finding the optimal dictionary \( \Phi \) that can minimize the overall reconstruction error:

\[
f(\Phi) = \sum_{i=1}^{N} ||x_i - \sum_{k=1}^{K} \Phi_k \alpha_{ik}||^2 + \lambda ||\alpha_i||_1, \forall i.
\]
Similarly, many other objective functions have also been defined with corresponding learning algorithms to achieved different goals in different applications, such as image compression, denoising, and image classification.

K-SVD [2] and its variants are widely used for dictionary learning. K-SVD learns a dictionary from a set of data points to minimize the overall reconstruction errors. It performs quite well for image denoising and compression. However, K-SVD is not a discriminative learning method and therefore it is not optimized for tasks like image classification or object recognition. Several K-SVD based discriminative dictionary learning algorithms have been proposed. Zhang et al. [57] incorporated the classification stage into the dictionary learning procedure to enforce its discriminative power. Jiang et al. [24] introduced a label consistent constraint in the learning procedure to enhance the discriminability. Lian et al. [26] presented probabilistic models to perform discriminative dictionary learning for image classification, and a general framework for supervised dictionary learning is reported. Zoltan et al. [45] proposed an online group-structured dictionary learning method with non-convex sparsity-inducing regularization to handle missing information, which gives excellent performance in image inpainting.

In this dissertation, we propose a novel selection-based dictionary learning algorithm. Instead of computing the dictionary columns, the selection-based dictionary learning algorithm aims to select the most representative samples from the training data to minimize the overall object function. The selection-based learning algorithm generates a more robust dictionary, since all dictionary components are selected from the truth set and will not introduce artificial components that may not fall into the target appearance subspaces. The detail of our proposed learning algorithm will be discussed in Chapter 3.

### 1.6 Contributions of the Dissertation

In this dissertation, we report on a set of studies involving sparse representations for robust visual tracking. This dissertation contributes to research as follows:

- A novel selection-based dictionary learning algorithm is proposed to achieve the best performance for tracking in order to discriminate a target from its background.
• A target voting method with encoded spatial information in the dictionary is a novel method to locate the target under large degrees of occlusion.

• A natural combination of static sparse dictionary and dynamic online updated basis distribution is proposed to balance adaptivity with stability of the procedure.

• The dictionary learning algorithm is also extended to general signal processing and other image applications, such as foreground detection and optical character recognition.

1.7 dissertation Overview

In Chapter 2 presents the two stage sparse learning and its application to visual tracking. The selection-based dictionary learning algorithm K-Selection is discussed in Chapter 3. The application of K-Selection to tracking is presented in Chapter 4 with a novel sparse tracker using a local appearance model. One of our tracking application for medical image analysis is also included in Chapter 5. The conclusion and future direction of the research are discussed in the last chapter.
Chapter 2
Two Stage Sparse Learning in Visual Tracking

2.1 Introduction

A learned sparse representation has been used for many application problems [19, 10, 31, 49] and successfully applied for tracking [35]. As discussed in Section 1.4, the likelihood of a given candidate sample being the target can be measured by the target’s optimal reconstruction error. L1 tracker[35] formulate the sparse representation problem as an $l_1$ minimization problem with non-negative constraints. It has been found to be efficient and adaptive to appearance changes, especially occlusion. However, several problems still arise with this approach:

- It is computationally expensive for very high dimensional data, which makes it unsuitable when using complex image features for fast tracking applications.

- The background pixels in the target templates do not lie on a linear subspace of templates. The scale of the reconstruction error for the background pixels is frequently greater than that for the target pixels, which can affect the accuracy of the sparse representation. It is therefore more reasonable to build a target template subspace from the pixels belonging to the target object.

- The non-negative constraints, while providing very good results when there are outliers, are vulnerable to complete tracking failures if the wrong templates are selected.

- Temporal correlation between target templates and spatial relations among adjacent image features are not considered.

- Since the sparse parameter $\tau$ in (1.3) has no physical meaning, it becomes difficult to tune this parameter.
We observed that the target can usually be represented by templates sparsely and only a subset of the features, which can discriminate the target from the background, are necessary to identify the target. Motivated by [35], but considering the challenges noted above, we propose a robust and efficient (fast) tracking algorithm with a two-stage sparse optimization procedure. The algorithm starts from feature selection by solving a dynamic group sparsity (DGS) [22] optimization problem. The DGS is then performed on the selected feature space for sparse reconstruction of the target. These two sparsity problems are optimized jointly and the final results are obtained by Bayesian inference. To our knowledge, this is the first study reporting a fast and robust tracking algorithm with a two-stage sparsity optimization. The contributions summarized in this chapter involve:

- A unified online updated sparse tracking framework which is designed to work with very high dimensional image features.
- The location of adjacent features and time adjacent target templates tend to be selected as a group in our sparse representation, which provides more robust tracking results.
- The sparse parameters do have physical meaning and therefore are more easily tuned.
- Pose variation, appearance changes, and heavy occlusions are handled by my algorithm.

2.2 Related Work

Online adaptive tracking methods have been intensively investigated recently. Grabner et al [17] update feature selection incrementally using training samples gathered from current tracking results, but this can lead to potential target drifting because of the accumulation of errors. Semi-online boosting [18] incrementally updates the classifier using unlabeled and labeled data together to reduce the likelihood of target drifting from such errors. Multiple Instance Learning boosting method (MILBoosting) [5] puts all samples into bags and labels them with bag labels. The positive (target) bag is required to contain at least one real positive, while the negative bags must contain only negative samples. The drifting problem is handled because a sample of the true target is thus learned implicitly by being included in the positive bag but not in the negative bag. The target is represented as a single online learned appearance model in incremental
visual tracking (IVT) [41]. But, since a single appearance model is unlikely to be sufficient to represent the target in a dynamic environment, multiple appearance models are incrementally learned during tracking in [56]. Online updating has proven to be a key component of adaptive tracking and is included in our algorithm.

A sparse representation is introduced for tracking in [35]. The target candidate is represented as a linear combination of the learned template set composed of both target templates and the trivial template which has only one non-zero element. The assumption is that good target candidates can be sparsely represented by both the target templates and the trivial templates. This sparse optimization problem is solved as an $l_1$ minimization problem with non-negative constraints. Another well known class of sparse optimization algorithms is iterative greedy pursuit. The earliest algorithms of this type have included matching pursuit [32] and orthogonal matching pursuit [46]. Subspace pursuit [12] and compressive sampling matching pursuit [38] were later designed to reach similar theoretical recovery guarantees as the Basis Pursuit problem while reducing computational complexity. However, the nonzero components of the
solution are not randomly distributed and tend to be clustered. Motivated by this observation, a dynamic group sparsity (DGS) recovery algorithm was introduced in [22]. The algorithm includes five main steps in each iteration: 1) pruning the residue estimation; 2) merging the support sets; 3) estimating the signal by optimizing the least square problem; 4) pruning the signal estimation and 5) updating the signal/residue estimation and support set. The algorithm is similar to that of SP/CoSaMP [12, 38] but, in addition, considers the effect of neighbors in the pruning process. DGS optimization has the additional advantage of yielding more robust result by forcing a group representation which can eliminate those templates that do not fall into the same linear subspace as its neighbors. In Figure 2.1 we show the group structure of the consecutive learned templates in one of our testing tracking sequences. The image features are projected onto a two-dimensional vector subspace, and clustered into six groups. The edges indicate the consecutiveness of templates. At the bottom of Figure 2.1, the plot of the sparse representation coefficients shows that the target is sparsely represented by two clusters of templates. In other words, if one of the templates in the group is selected, its temporally adjacent templates tend to be selected also in our sparse representation using DGS.

2.3 Bayesian tracking framework

Let affine parameters $\chi_t = (x, y, s, r, \theta, \lambda)$ represent the target state in the $t$-th frame, where $x$ and $y$ are the coordinates, $s$ and $r$ are the scale and the aspect, $\theta$ is the rotation angle, $\lambda$ is the skew. The tracking problem can be formulated as an estimation of the state probability $p(\chi_t | z_{1:t-1})$, where $z$ represents observations from the previous $t - 1$ frames. Sequential Bayesian tracking based on a Markovian assumption estimates and propagates the probability of the state by recursively performing predictions by integration over the prior $t-1$ possible states

$$p(\chi_t | z_{1:t-1}) = \int p(\chi_t | \chi_{t-1}) p(\chi_{t-1} | z_{1:t-1}) d\chi_{t-1} \quad (2.1)$$

and updating the prediction for the $t$-th state

$$p(\chi_t | z_{1:t}) \propto p(z_t | \chi_t) p(\chi_t | z_{1:t-1}). \quad (2.2)$$
Algorithm 1: Tracking with two stage sparsity optimization

**Input:** Target’s initial state \( \chi_0 \), sparsity parameter \( K_0 \) for feature selection, \( K_1 \) and \( K_2 \) for target templates and trivial templates, respectively.

**Initialize:** Construct \( n \) training samples \( \{X \in \mathbb{R}^{n \times p}, L \in \mathbb{R}^{n \times 1}\} \), where \( X \) is the sample matrix, \( L \) is the label and \( p \) is the dimension of the feature vector.

1. For each frame \( t = 1 : T \) in the video where \( T \) is the total number of frames:
   2. Perform DGS to solve \( w^* = \arg\min_w ||Xw - L||_2 \), subject to: \( |w|_0 \leq K_0 \) (when \( t = 1 \) we will use the initializations).
   3. Construct diagonal matrix \( W \), \( W_{i,i} = \begin{cases} 1, & w^*_i \neq 0 \\ 0, & \text{otherwise} \end{cases} \)
   4. Generate \( N \) candidate samples \( y_i \) in state \( \chi^i_t \).
   5. For each \( y_i, i = 1 : N \)
      6. Let \( W' \in \mathbb{R}^{K_0 \times p} \) as the matrix contains all non-zero rows of \( W \),
      7. \( \Phi' = W'\Phi, y'_i = W'y_i, \) and \( f' = W'f, \)
      8. perform DGS to solve \( (\alpha^*, f^*) = \arg\min_{\alpha,f} \left| \left| \Phi'W' \left[ \begin{array}{c} \alpha \\ f \end{array} \right] - y'_i \right| \right|_2 \), subject to: \( ||\alpha||_0 \leq K_1 \), \( ||f||_0 \leq K_2 \).
      9. \( \epsilon_i = ||\Phi'\alpha^* - y'_i||_2. \)
     10. \( p(z_t|\chi^i_t) = \exp(-\epsilon_i). \)
    12. end for
   13. \( \chi^*_t = \arg\max_{\chi_t} p(\chi_t|z_{1:t}). \)
   14. Update the training set and template library with tracking results.
   15. end for

The transition model \( p(\chi_t|\chi_{t-1}) \) can be constrained and made mathematically tractable by assuming a Gaussian distribution \( \mathcal{N}(\chi_t|\chi_{t-1}, \sigma) \). The observation model \( p(z_t|\chi_t) \) represents the likelihood of \( z_t \) being generated from state \( \chi_t \).

In our algorithm, \( N \) candidate samples are generated based on the state transition model \( p(\chi_t|\chi_{t-1}) \). Again, for tractability and simplicity, the state variables are considered to be independent of each other. Each candidate sample \( I_i \) with state \( \chi^i_t \) is reconstructed from the template library \( \Phi \) using the dynamic group sparsity (DGS) approach outlined above. The likelihood \( p(z_t|\chi^i_t) = \exp(-\epsilon_i) \) where \( \epsilon_i = \min_{\alpha} ||\Phi\alpha - I_i|| \) is the optimized reconstruction error of \( I_i \) and \( \alpha \) represents the sparse coefficients. Instead of solving the optimization problem in the full feature space, we perform the sparse optimization in a selected feature space with high discriminative power. This enables us to use some advanced high dimensional features, such as Haar-like feature, without sacrificing the efficiency of the algorithm. The final result is obtained by maximizing \( p(\chi_t|z_{1:t}) \). Once the tracking state is found, new samples are extracted and used to update the training set and the template library by the online method.
2.4 Two-stage sparse representation

Given the learned target template library \( \Phi \in \mathbb{R}^{p \times m} \), where \( m \) is the number of templates and \( p \) is the dimension of the features, let \( \Phi_1 = [\Phi, I] \) and \( \alpha_1 = \begin{bmatrix} \alpha \\ f \end{bmatrix} \), where \( \alpha \) represents the sparse coefficient vector and \( f \) denotes the occlusion, the candidate sample \( y \) is sparsely reconstructed from \( \Phi \) by minimizing the \( l_2 \) errors and finding \( \alpha \) with \( K_1 \) nonzero components and \( f \) with \( K_2 \) nonzero components using the greedy method:

\[
\alpha_1 = \underset{\alpha, f}{\text{argmin}} \| \Phi_1 \alpha_1 - y \|_2, \quad \text{while } \| \alpha \|_0 \leq K_1 \text{ and } \| f \|_0 \leq K_2.
\] (2.3)

Equation (2.3) can be solved efficiently when the dimension of the feature space and candidate search space are small. However, it is computationally expensive for very high dimensional data, which makes it unsuitable if complex image features are used. Because only some of the features, which can discriminate the target from the background, are necessary to identify the target, we argue that the effective dimension of the feature space can be decreased to \( K_0 \) dimension with a diagonal matrix \( W \) such that the number of nonzero components in \( W \) is no greater than \( K_0 \). The \( i\)-th feature is activated if \( W_{ii} \) is nonzero. Given \( n \) available samples \( X \in \mathbb{R}^{n \times p} \) and their labels \( L \in \mathbb{R}^{n \times 1} \), the joint sparse solution can be found as:

\[
(\alpha_1, W) = \underset{\alpha_1, W}{\text{argmin}} \lambda \| W \Phi_1 \alpha_1 - W y \|_2
+ \beta F(W, X, L) + \tau_1 \| \alpha_1 \|_1 + \tau_2 \| \text{diag}(W) \|_1
\] (2.4)

where \( F(W, X, L) \) is the loss function in the selected feature space for the training dataset and samples in the current frame, and \( \tau_1 \) and \( \tau_2 \) are the sparse parameters. As we explained earlier, the parameters \( \tau_1 \) and \( \tau_2 \) in (2.4) have no physical meaning and therefore it is difficult to tune their values. In our procedure, we apply the greedy algorithm to directly solve the original \( l_0 \) minimization problem for a sparse representation. In this way (2.4) can be rewritten as:
\[(\alpha_1, W) = \arg \min_{\alpha_1, W} \lambda \| W \Phi_1 \alpha_1 - W y \|_2 + \beta F(W, X, L), \]

subject to: \[\| \text{diag}(W) \|_0 \leq K_0, \| \alpha \|_0 \leq K_1 \text{ and } \| f \|_0 \leq K_2. \] (2.5)

As it is hard to find an optimum solution for (2.4) when both \(\alpha_1\) and \(W\) are unknown, we solve (2.5) using two stage dynamic group sparsity optimization with a greedy method. The first stage is to select the sparse set of features that are most discriminative in separating the target from background. Then the generative likelihood of each sample is estimated in the second stage with the sparse representation. The details of the algorithm are shown in Algorithm 1. We will explain each stage in the following sections.

2.4.1 Feature selection

Given a set of training data \(X = \{x_i \in \mathbb{R}^{1 \times p}\}\) with \(L = \{l_i\}, i = 1 \ldots n\) as the labels, the term \(F(W, X, L)\) in equation 2.4 is defined as

\[F(W, X, L) = e^{-\sum_{i=1}^{n}(x_i w)l_i},\] (2.6)

where \(w \in \mathbb{R}^{p \times 1}\) is a sparse vector. The \(j\)-th feature is selected if \(w_j \neq 0\). The solution that minimizes \(F(W, X, L)\) can be found by solving the following sparse problem

\[w^* = \arg \min_w \|Xw - L\|, \text{ subject to: } \|w\|_0 \leq K_0\] (2.7)

where \(K_0\) is the maximum number of features that will be selected. Here we want to emphasize that using a greedy method for optimization, the parameter \(K_0\) does have physical meaning it corresponds to the number of features we plan to select. Considering Haar-like features, we do have the spatial relationship between neighborhood features. For example, if a small patch is occluded, the features extracted from this region will tend to be treated as a group in the sparse optimization. Let \(N_w(i,j)\) as the value of \(j\)-th neighbor of \(i\)-th feature, the support set
is pruned based on $Z$

\[
z_i = w_i^2 + \sum_{j=1}^{\tau} \theta_j^2 N_w(i, j), \quad i = 1 \ldots p
\]  

(2.8)

in DGS taking the neighborhood relationship into consideration, where $\theta$ is the weight of neighbors. With the optimal $w$ found by DGS, The diagonal matrix $W$ can be constructed as

\[
W_{j,j} = \begin{cases} 
1, & w_j^* \neq 0 \\
0, & \text{otherwise;}
\end{cases}
\]  

(2.9)

Benefiting from the sparse solution to (2.7), we will be able to use complex, or advanced high dimensional features without sacrificing the efficiency of the algorithm. The other benefit is the object selection in the target region. The target templates usually contain some background features which are not linear. By doing discriminative feature selection, features from background pixels in the target templates are eliminated. The target template library is therefore more efficient and robust.

### 2.4.2 Sparse Reconstruction

After we calculate the weighting matrix $W$, the $\alpha$ and $f$ in equation (2.4) can be found in the second stage

\[
(\alpha, f) = \arg\min_{\alpha, f} \| W_1 \alpha - W y \|, \text{ subject to: } \|\alpha\|_0 \leq K_1 \text{ and } \|f\|_0 \leq K_2.
\]  

(2.10)

where $\Phi_1 = [\Phi, I]$ and $\alpha_1 = \begin{bmatrix} \alpha \\ f \end{bmatrix}$. Let $W' \in \mathbb{R}^{K_0 \times p}$ as the matrix contains all nonzero rows of $W$. We define $\Phi' = W' \Phi$ and $y' = W' y$. Note that in this step we already reduced the feature dimension from $p \times m$ to $K_0 \times m$ where $m$ is the number of templates in the target library. At this stage the following equation is solved

\[
(\alpha^*, f^*) = \arg\min_{\alpha, f} \left\| \begin{bmatrix} \alpha \\ f \end{bmatrix} \right\|_2, \text{ subject to: } \|\alpha\|_0 \leq K_1 \text{ and } \|f\|_0 \leq K_2.
\]  

(2.11)
Here the sparsity parameters $K_1$ and $K_2$ have a clear physical meaning, since $K_1$ controls the sparsity of the target template representation and $K_2$ controls the tolerance of occlusion. Then the likelihood of the testing sample $y$ being a target is $e^{-\|\Phi'\alpha^* - y\|^2_2}$ and final result is obtained by maximizing the $p(x_t|z_{1:t})$.

As we have already shown in Figure 2.1, the target templates exhibit a group structure and the temporally consecutive templates are likely to fall into the same group. The correct target sample can be reconstructed by sparse grouped templates. In our algorithm, we take into consideration the relationship between the template neighbor, and tend to select grouped templates. This leads to a sparse vector overall (globally), but which is dense in terms of its locally grouped consecutive templates. The $l_1$ minimization algorithm with non-negative constraints in [35] provides a very sparse representation in the template reconstruction coefficients, but it is vulnerable to outliers, namely, one single mistake in a template library can lead to complete tracking failure. For example, if a background sample is added into the template incorrectly, in an static background, it probably will have high matching likelihood since they are static most of time and can often find the perfect reconstruction. We avoid this problem in our algorithm by forcing a group selection of sparse coefficients. Since the outlier template is not in the same linear space as its neighbors, this can prevent it from being selected as it will lead to large reconstruction errors where even a standalone matching has a high score.

Once the tracking result is confirmed, the template library is incrementally updated as in [35]. The samples with high likelihood which are nearer to the target are added to the training set as positive samples, while the others are added as negative samples. This procedure is repeated for each frame in a whole sequence. The joint optimization of the two stage sparsity problem thus provides a fast, robust and accurate tracking result.

### 2.5 Experimental Results

The proposed tracking algorithm has been evaluated using five challenging sequences with 3217 frames in total. The method is compared with three of the latest state-of-the-art tracking methods, namely L1 tracker(L1) [35], Incremental Visual Tracking (IVT) [41], and Multiple
Figure 2.2: The tracking results of a car sequence in an open road environment. The vehicle was driven beneath a bridge which produced considerable changes in illumination. Results are shown for our algorithm, MIL, L1, and IVT are given in the first, second, third, and fourth row, respectively.

Instance Learning (MIL) [5]. The tracking results of the algorithms being compared are obtained by running the binaries or source code provided by their authors using the same initial positions. The source code of L1, IVT, MIL can be obtained from the following URLs: 1 2 3. The first, second, third and fourth sequences were obtained from [41], and the fifth sequence was downloaded from [5].

In Section 2.5.1 I present the visual evaluation of the comparative tracking results. Several frames in five sequences are shown in the figures. Detailed quantitative evaluation of the comparative tracking are presented in Section 2.5.2. The tracking error-time graphs for four sequences are plotted. Both visual and quantitative results demonstrate that our method provides more robust and accurate tracking results.
2.5.1 Visual Evaluation of Comparative Experimental Results

The first sequence was captured in an open road environment. The tracking results based on the 4, 56, 200, 238, 309 frames are illustrated in Figure 2.2. The L1 algorithm starts to show some drifting on the 56-th frame. The MIL starts to show some target drifting from the 200-th frame, and eventually loses the target at the 238-th frame. IVT can track this sequence quite well, as it was by our proposed algorithm during the entire sequence.

The second sequence contains a moving face which must be tracked. The 2, 47, 116, 173, and 222 frames are shown in Figure 2.3. The L1 algorithm fails to track the target when there are both pose and scale changes, shown in the 116-th frame. The MIL method can roughly capture the position of the object, but does have some target drift problems, especially in the 173-th and 222-th frame. Our proposed two stage sparse tracking algorithm can track the moving face accurately through the whole sequence while the IVT produces some errors, especially on the 222-th frame.

1 http://www.ist.temple.edu/~hbling/code_data.htm
2 http://www.cs.toronto.edu/~dross/ivt/
3 http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml
Figure 2.4: The tracking results of a plush toy moving around under different pose and illumination conditions. The order of the rows is the same as in Figure 2.2.

The third image sequence is shown for frame 2, 241, 459, 611, and 693 in Figure 2.6. The L1 method starts to exhibit a drifting problem from roughly the 200-th frames, shown in the 241-th and 459-th frame. The MIL algorithm provides very good tracking results in this sequence. IVT fails to follow the target-object on the 611-th frame after major pose variation and can not be subsequently recovered. Our algorithm, in contrast, provides robust and accurate tracking results for this long sequence sequence.

In the fourth sequence, the vehicle was driven in a very dark environment and captured from another moving vehicle. The 2, 35, 208, 239, 378 frames are presented in Figure 2.3. The L1 algorithm starts to fail in tracking the target from the 35-th frame. The MIL can roughly capture the position of the object at the start, but develops target drifting from the 208-th frame distracted by light. IVT can track the target through the whole video sequence but it is not as accurate as our results, which can be found in the 378-th frame.

The results of the fifth sequence are shown in Figure 2.6. In this sequence we show the robustness of our algorithm in handling occlusion. The frame indexes are 10, 427, 641, 713, and 792. Starting from the 641-th frame, our method perform consistently better compared with the other methods.
Figure 2.5: The tracking results of the car sequence in a dark environment. This sequence has low resolution and poor contrast, which introduce some landmark ambiguity. The order of the row sequences is the same as in Figure 2.2.

2.5.2 Quantitative Evaluation of Comparative Experimental Results

For fair comparison, the tracking error $e$ in each frame is measured as $e = \epsilon / d$, where $\epsilon$ is the offset of center from the ground truth and the $d$ is the diagonal length of the target rectangle. For perfect tracking, the $e$ should be equal to zero for each frame. In Table 5.1, we compared the quantitative $e$ using our proposed algorithm with L1, MIL and IVT. The best result in each column is shown in bold in Table 2.1. The missing column represents the number of frames where the $e > 1$. For a fair comparison, we do not count these failing frames when computing the overall mean and variance in the 7-th and the 8-th columns in Table 5.1. Measured by the public open benchmark, on average our algorithm only has 7% of drifting errors and never misses one single frame in the five tracking sequences which contain thousands of frames in total. In Figure 2.7 we present the tracking error-time curve. We can see that except for the fifth sequence, in which we obtain similar results as IVT (IVT will intend to shrink the window to very small size but won’t lose the center of the target, as shown in Figure 2.6), our algorithm outperforms the other methods. The method is computationally efficient. Even using a MATLAB implementation, it can process two frames/second.
2.6 Conclusion

We have proposed an online robust and fast tracking algorithm using a two-stage sparse optimization approach. No shape or motion priors are required for this algorithm. Both the training set and the template library models are updated online. Two stage sparse optimization is solved jointly by minimizing the target reconstruction error and maximizing the discriminative power by selecting a sparse set of features. The experimental results demonstrate the effectiveness of

<table>
<thead>
<tr>
<th>Mean</th>
<th>L1</th>
<th>MIL</th>
<th>IVT</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq1</td>
<td>1.10</td>
<td>1.02</td>
<td>0.04</td>
<td>0.03</td>
</tr>
<tr>
<td>Seq2</td>
<td>1.31</td>
<td>0.34</td>
<td>0.09</td>
<td>0.08</td>
</tr>
<tr>
<td>Seq3</td>
<td>0.89</td>
<td>0.17</td>
<td>1.15</td>
<td>0.16</td>
</tr>
<tr>
<td>Seq4</td>
<td>3.22</td>
<td>1.16</td>
<td>0.07</td>
<td>0.08</td>
</tr>
<tr>
<td>Seq5</td>
<td>0.21</td>
<td>0.12</td>
<td>0.13</td>
<td>0.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overall</th>
<th>L1</th>
<th>MIL</th>
<th>IVT</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.38</td>
<td>0.31</td>
<td>0.08</td>
<td>0.07</td>
</tr>
<tr>
<td>Variance</td>
<td>0.26</td>
<td>0.29</td>
<td>0.08</td>
<td>0.06</td>
</tr>
<tr>
<td>Median</td>
<td>1.34</td>
<td>0.46</td>
<td>0.05</td>
<td>0.04</td>
</tr>
<tr>
<td>Max</td>
<td>5.09</td>
<td>3.82</td>
<td>5.82</td>
<td>0.34</td>
</tr>
<tr>
<td>Missing</td>
<td>828</td>
<td>55</td>
<td>470</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.1: The overall quantitative tracking performance comparison of proposed robust tracking method with two stage sparse optimization, L1 [35], MIL [5], and IVT [41].
Figure 2.7: The tracking accuracy $e$ for each frame in four different sequences.

our method for handling a number of challenging sequences.
Chapter 3
Selection-based Dictionary Learning

3.1 Motivation

As we discussed in Chapter 1, the existing dictionary algorithms have been used in many applications and demonstrated their advantages. However, there are still some disadvantages of those existing dictionary learning algorithms in the application to visual tracking.

- The components in the dictionaries are computed to minimize the overall reconstruction error of the desired target-object. It is very likely to generate some components that can compensate for the reconstruction residuals but do not fall into the subspaces spanned by the target appearances. These additional components will contribute to potential tracking error and lead to drifting problems.

- The dictionary components generated by the existing algorithms do not have a clear physical meaning in terms of the image or target-object appearance.

- The dictionary learned from the existing algorithms are too general, so that they can also been used to represent other objects as well, such as background samples also, showing that they are economically not effective.

- The optimization procedure for most of these existing algorithms are computation expensive.

Considering the applications of sparse representation in tracking, we proposed a selection-based dictionary learning algorithm which can generated suboptimal solution of the object function of dictionary learning. The selection-based dictionary learning algorithm constrains the generality of the dictionary and lead to better performance in visual tracking which need to differentiate the target samples from the background samples.
3.2 Selection-based Dictionary Learning

3.2.1 Objective function

We assumed that the dictionary $\Phi$ is given in all the above. Many methods have been proposed to learn a dictionary to minimize the overall reconstruction error in sparse representations [13, 30, 2]. The target templates are stored and updated to form the dynamic dictionary in [35, 28]. Here, we introduce a new method to learn the dictionary as a basis selection by gradient descent. Given a dataset $X = \{x_i | i = 1 \ldots N\}$, the problem can be formulated as selecting $K$ data vectors from the dataset as the dictionary components to minimize the objective function:

$$f(\Phi) = \sum_{i=1}^{N} ||x_i - \sum_{k=1}^{K} \Phi_k \alpha_{ik}||^2 + \lambda ||d_i \odot \alpha_i||^2,$$

s.t. $1^T \alpha_i = 1, \forall i$. \hspace{1cm} (3.1)

where $\Phi_k = x_{b_k}$ and $b_k$ is the index of data vector selected as the $k$-th basis vector. The $d_i$ is exponential to the distance of $x_i$ to the dictionary. $\alpha_i$ is the vector of representation coefficients. In terms of visual tracking, the dictionary selection problem can be described as selecting a subset of the given target templates to minimize the reconstruction error over all training data. The goal is to select the most representative templates that are general enough to represent the appearance of all other target samples. Exhaustive search would be needed to find the optimal solution, so we propose instead a more efficient method which can converge to a suboptimal solution, called $K$-Selection.

3.2.2 K-Selection

Similarly to other local algorithms, K-Selection first selects an initial subset of the training data as the initial dictionary. It then updates the selected subset iteratively similarly to gradient decent algorithms. However, the dictionary components cannot be moved arbitrarily along the gradient direction to minimize the objective function. Instead, K-Selection selects a new dictionary component to replace the current ones so as to yield smaller reconstruction errors.
iteratively.

3.2.3 Initial Dictionary Selection

An appropriate initialization is very critical for local algorithms. The intuition when selecting initial dictionary components for a sparse representation is to select the components that will be used heavily in representing other data points in the training set.

Sparse Graph

With all data points $x_i \in \mathbb{R}^p$ represented in $p$ dimensional space, we can construct a sparse directed graph $\langle V, E \rangle$. $V$ includes all the data points as the vertices of the graph, while $E_{i,j}$ is the directed edge from data points $x_i$ to $x_j$. The weight $\omega_{i,j}$ of edge $E_{i,j}$ measures the importance of data point $x_j$ for representing $x_i$. The importance of a single data point $x_j$ to the whole training data at can be measured by the sum of the weights of all edges pointing to $x_j$.

The initial set of basis vectors can be chosen by the following steps:

Step 1: For any data point $x_i \in X$, the sparse representation with all other points as a dictionary can be solved by

$$
\min_{\omega_i} ||x_i - \sum_{j=1, j \neq i}^{N} x_j \omega_{ij}||^2 + \lambda ||d_i \odot \omega_i||^2,
$$

$$
s.t. 1^T \omega_i = 1,
$$

(3.2)

where $\omega_{ij}$ indicates the importance of the $j$-th data point for sparsely representing the $i$-th data point as shown in Figure 3.1

Step 2: The importance of the $j$-th data point, $w_j$, to be selected as a basis vector is the weighted sum of all edges pointing to $x_j$.

$$
w_j = \sum_{i=1}^{N} |\omega_{ij}| e^{-\epsilon_i^2/\sigma^2}.
$$

(3.3)

The reconstruction error $\epsilon_i$ indicates the value of this representation. In other words, the importance of the $j$-th data point is its weighted contribution in representing the entire dataset.
Step 3: The first $K$ data vectors with the largest weights $w$ are selected as the initial basis.

### 3.2.4 Iterative Optimization

After the initialization, a new data vector will be selected to replace the $t$-th basis vector in order to minimize the cost function iteratively. Let $\alpha_i$ be the solution of locality-constrained linear coding for representing $x_i$ using the current dictionary $\Phi$. The dictionary is then updated to fit the dataset without the locality constraint. The gradient with respect to the $t$-th basis can be approximated as

$$\nabla f_t = \frac{\partial f}{\partial \Phi_t} = -2 \sum_{i=1}^{N} (x_i - \sum_{k=1}^{K} \Phi_k \alpha_{ik}) \alpha_{it}. \quad (3.4)$$

As shown in Figure 3.2, instead of directly moving the basis toward the direction of the negative gradient $r_t = -\nabla f_t$, we perform the update by selecting the data point $x_l$ which has the largest correlation between the displacement and $r_t$

$$COR(x_l, x_{b_t}, r_t) = \frac{(x_l - x_{b_t})^T r_t}{\| (x_l - x_{b_t}) \|_2 \| r_t \|_2}. \quad (3.5)$$

The data point $x_l^*$ with the maximal value of $COR$ is selected as a potential candidate to
replace the $t$-th basis. Let $f_{\min}$ be the current residual and $f_{rep}$ the residual after replacing the $t$-th basis with $x^*_t$, then the replacement will be carried out only if $f_{\min} > f_{rep}$.

### 3.3 Applications Exploration

#### 3.3.1 Target Modeling in Visual Tracking

Compared to other dictionary learning methods, such as $K$-SVD, the dictionary learned with $K$-Selection has a constrained capability to represent the dataset. However, the target library learned with $K$-SVD is so general that some of the background image patches can also turn out to be well represented. This is not desirable in visual tracking, which requires strong discriminative ability. In order to provide higher discriminative power, we limit the space spanned by the learned target library strictly to the target model itself, by directly selecting the $K$ data vectors from the dataset.

**Reconstruction vs. Discriminative Power**

This section describes the performance of the proposed $K$-Selection dictionary learning method in terms of its generative and discriminative capability. As noted in this section, it is worthwhile to evaluate not only the reconstruction error, but also the discriminative power for a sparse dictionary learning method. We claim that under certain conditions, discriminative power counts more than actual overall reconstruction error. In this set of experiments, more than a hundred
thousand image patches were extracted from the target region, and the same number of background patches were randomly generated from the regions outside the target. The dictionaries were trained using the target patches extracted from the first frame only.

If $X^+$ and $X^-$ are the set of target patches and background patches, the reconstruction error is measured as

$$E(X) = \frac{1}{N} \sum_{i=1}^{N} ||x_i - \Phi \alpha_i^*||$$

(3.6)

where $\alpha_i^*$ is the sparse solution of the $i$-th patch using Eqn. (1.6). The difference between $E(X^+)$ and $E(X^-)$ is used to measure the discriminative power of the learned dictionary. A larger difference $|E(X^+) - E(X^-)|$ indicates stronger discriminative power.

The popular $K$-SVD method [2] with orthogonal matching pursuit (OMP) [46] is used for comparison. As shown in Figure 3.3(a), it is not surprising that the dictionary learned with $K$-SVD has a smaller overall reconstruction error than our $K$-Selection method as it explicitly minimizes the $l_2$ reconstruction errors. However, the dictionary learned by $K$-SVD with OMP can also represent the background patches, which leads to relatively weaker discriminative power compared to our $K$-Selection algorithm, as shown in Figure 3.3(b). Therefore, $K$-Selection exhibits larger reconstruction errors but stronger discriminative power, which makes it more suitable for tracking and some other applications. Also, discriminative power does not always increase by adding more basis patches, because the added basis will contribute to the reconstruction of both background and target. We have experimentally found that a very small basis set is often sufficient to discriminate target from background.
Figure 3.3: Performance of the learned dictionaries using $K$-Selection (red) and K-SVD (blue dash) evaluated by reconstruction errors (a) and discriminative power (b).
Chapter 4

Robust tracking using a local sparse appearance model

4.1 Introduction

All the methods we have discussed in previous chapters try to model the target as a single entity, and therefore cannot handle partial occlusion very well. Fragment-based tracking presented in [1] coupled with a voting map can accurately track a partially occluded target. However, this method tracks each target patch with a static template, which limits its expressive power. It may fail in a dynamic environment which exhibits appearance changes or pose variations.

In this chapter, we propose and test a robust tracking algorithm with a local sparse appearance model (SPT) and \(K\)-Selection dictionary. The algorithm’s key components are a static sparse dictionary which is used to limit drifting and preserve the flexibility in its linearly spanned subspace; a dynamic dictionary basis distribution is represented by a sparse coding histogram and is updated online; a sparse representation based voting map and reconstruction error regularized mean-shift are used to finally locate the center of the target-object. Figure 4.1 illustrates an overview of the proposed algorithm. To our best knowledge, it is the first publication in the literature for tracking with an online learned fragment-based sparse representation using a static basis but a dynamic structure, while \(K\)-selection is the first selection based dictionary learning method for sparse representation. The contributions of this work are:

- A natural combination of static sparse dictionary and dynamic online updated basis distribution considering both adaptivity and stability.
- A sparse representation based voting map and sparse constraint regularized mean-shift for object tracking.
4.2 Tracking Framework

In this chapter, I present a new tracking algorithm using local appearance models learned with K-Selection. The tracking framework is illustrated in Figure 4.1. From a given target at the first frame, a set of target patches can be generated with a sliding window to generate training data. Then the most representative subset of the generated target patches is selected using K-Selection.

With the selected dictionary, a sparse coding histogram and spatial configuration can be learned from the target template provided at the training stage.

For the current frame in a given video, the likelihood of each image patch being a target patch can be measured using sparse representation as described in Section 1.4. In this work, we use LLC to solve the sparse representation of each candidate sample. Combining the likelihood and encoded spatial information, the target center can be found by voting and sparse constrained mean-shift. With the mean-shift, it is not necessary to solve the sparse representation problem for all image patches. The estimated target center will gradually move to the new location. Only the sparse representation of those image patches within the moving area need to be solved. Thus the algorithm is efficient and can be parallelized for real-time performance.

4.3 Target Modeling

4.3.1 Sparse Coding Histogram

With the sparse representation model presented above, the local patches from target regions should have smaller reconstruction errors than those from background clutter. But as we show in Figure 4.1(e), some regions of the book, which belong to the background, get assigned a relatively large probability since their appearance falls within the subspace spanned by the target dictionary basis. A few contaminated bases selected from the background due to an inaccurate initial rectangle may affect the performance of tracking, especially in the static camera scenario where target appearance changes but the background remains static.

Thus, more structure information is necessary for accurately identifying the target. In this section, we propose a sparse coding histogram to represent the appearance distribution of the
Figure 4.1: The target appearance (a) is modeled with a dictionary (b) and a sparse coding histogram (c). The confidence map (e) of the image (d) is the inverse of the reconstruction error from the learned target dictionary. The target center is found by voting and sparse constraint with regularized mean-shift on the probability map (f).

For each image patch $x_i$ in the template, let $\alpha_i$ be the optimal sparse coefficient vector in Equ. (1.6). The coding histogram is defined as the coefficient of the sum of the coefficients of the basis with non-zero coefficient values.

$$H(b_j) = \sum_{i=1}^{N} |\alpha_{ij}|.$$  (4.1)
Similarly to color histograms, a sparse coding histogram indicates how the appearance dictionary basis is distributed over the target model.

**Target Model:** Let the target center be taken as the origin of the model frame of reference. Define $x_i, i = 1: N$ as the vectorized image patches centered at pixel position $c_i$, an isotropic kernel $k(c_i)$ is applied to assign smaller weights to pixels far away from the center. The value of the $j$-th bin $q_j$ in the target model can be computed as a weighted sum:

$$q_j = C \sum_{i=1}^{N} k(||c_i||^2)|\alpha_{ij}|$$  \hspace{1cm} (4.2)

where $C$ is a normalization constant to ensure $\sum_{j=1}^{K} q_j = 1$, and $\alpha_{ij}$ is the $j$-th coefficient of the $i$-th image patch.

**Target Candidate:** Define $x_i^*, i = 1: N'$ as the vectorized image patches centered at pixel position $c_i$ inside the window centered at $y$. The value of the $j$-th bin, $\hat{p}_j(y)$, in the candidate model can be computed as:

$$\hat{p}_j(y) = C \sum_{i=1}^{N'} k(||\frac{y - c_i}{h}||^2)|\alpha_{ij}^*|$$  \hspace{1cm} (4.3)

where $\alpha^*$ is the solution of Eqn. (1.6) and $h$ is the scale factor.

The sparse coding histogram is dynamic when a target experiences changes and is updated online. Let $y$ be the new target center found for the current frame and $\hat{p}_j(y)$ its coding histogram from Eqn. (4.3), the new appearance basis histogram can be updated with learning rate $\gamma$:

$$q'_j = q_j(1 - \gamma) + \hat{p}_j(y)\gamma$$  \hspace{1cm} (4.4)

### 4.4 Target Localization

**4.4.1 Sparse Constraint Regularized Mean-shift**

In this section we present an iterative tracking algorithm to locate a target with a local appearance model. Let $y$ be the candidate target center, while $X = \{x_i, i = 1...N\}$ represents $N$
patches in the window $W$ centered at $y$. Tracking is aimed at locating the target with maximum generative likelihood, and match the target model and candidate models.

The probability of $y$ being the target center can be estimated by the the products of the probability of each candidate patch within $W$ as potential target patches,

$$P(y|\Phi) = C \prod_{i=1}^{N} e^{-k(||y-c_i||^2 \sigma^2)^2}. \quad (4.5)$$

where $\epsilon_i$ is the sparse reconstruction error for the $i$-th patch. The log likelihood of the target candidate is then:

$$L(y|\Phi) = \sum_{i=1}^{N} -k(||y-c_i||^2 \sigma^2)^2 \epsilon_i^2. \quad (4.6)$$

The Bhattacharyya metric is used to measure the distance between the sparse coding histograms of the target and candidate models

$$d(y) = \sqrt{1 - \rho(\hat{p}(y), q)}, \quad (4.7)$$

$$\rho(\hat{p}(y), q) = \sum_{j=1}^{K} \sqrt{\hat{p}_j(y)q_j}. \quad (4.8)$$

Considering that the similarity of the target to the learned dictionary and the similarity to the dictionary basis distribution are independently conditioned on the same candidate, the overall objective function can be formulated as:

$$\hat{\rho}(y, \Phi) = \sum_{j=1}^{K} \sqrt{\hat{p}_j(y)q_j L(y|\Phi)}. \quad (4.9)$$

The first component in $\hat{\rho}(y, \Phi)$ measures the match between the distribution of the target model and candidate model, while the second term measures the probability of the candidate being generated from its target library $\Phi$. 
Assume that we have an initial guess of the center as $y_0$. Using a Taylor expansion, Eqn. (4.9) can be rewritten as:

$$
\hat{\rho}(y, \Phi) \approx -\frac{1}{2} \sum_{j=1}^{K} \sqrt{\hat{p}_j(y_0)} q_j L(y_0|\Phi) \\
+ \sum_{j=1}^{K} \sqrt{\hat{p}_j(y_0)} q_j L(y|\Phi) \\
+ \frac{1}{2} \sum_{j=1}^{K} \hat{p}_j(y) \sqrt{\frac{q_j}{\hat{p}_j(y_0)}} L(y_0|\Phi) \\
= C_1 + \frac{1}{2} \sum_{i=1}^{N} w_i k(||y - c_i||^2),
$$

(4.10)

$$
w_i = \sum_{j=1}^{K} \sqrt{\hat{p}_j(y_0)} q_j \left( \frac{L(y_0|\Phi)|\alpha_{ij}|}{\hat{p}_j(y_0)} + \frac{-2\epsilon^2}{\sigma^2} \right)
$$

(4.11)

where $C_1$ in $\hat{\rho}(y, \Phi)$ does not depend on $y$. The second term in Eqn. (4.10) has to be maximized to minimize the Bhattacharyya distance. It also represents the density estimation computed with kernel $k(.)$ at $y$ with weight $w_i$. The target center can be found iteratively using Mean-shift.

$$
\hat{y} = \frac{\sum_{j=1}^{N'} c_i w_i g(||y_0 - c_i||^2)}{\sum_{j=1}^{N'} w_i g(||y_0 - c_i||^2)}
$$

(4.12)

To measure the size of the target, the tracking procedure can be carried out with several scale values and the target center and scale with the maximum of Eqn. (4.9) selected as the tracking result. The motion and scaling are assumed to be continuous.

4.4.2 Voting in a Sparse Representation

For accurately locating the target center in an occlusion scenario, a new voting method is proposed to improve the robustness of our tracker. Traditional fragment-based or part-based tracking methods track each part of the target and use the relative position of the target center to vote on a final tracking result. However, locating a large number of fragments is time consuming and
Training Stage: A set of target patches inside the target region is extracted as the training data. Using the learned dictionary with $K$-Selection, the sparse coefficients $\alpha_i$ for each training patch $x_i$ can be solved using LLC. For a given dictionary patch $\Phi_j$, its probability map for being a relative target center $d_c(dx, dy)$ can be computed by:

$$P_c(d_c, j) = P_c(d_c, j) + \alpha_j^2 k(||d_c/h||^2),$$  \hspace{1cm} (4.13)

where $dx$ and $dy$ represent the position of target center relative to the $i$th patch’s center. The $k(x)$ is a Gaussian kernel that minimizes the effect of background patches to be included in the target bounding box.

An example of the probability map of the relative target center for one given dictionary patch is shown in Figure 4.2. The dictionary patch is labeled using the green rectangle shown in the left image. All image patches labeled with red rectangles use the selected basis (with non-zero coefficient) for the sparse representation. Each of them contributes to the probability map of the basis weighted by the coefficients in Eqn. (4.13). Let the position of the dictionary patch
Algorithm 1: Compute sparse representation based voting

Define: $x_i$ as the $i$th image patches centered at position $c_i$.

1. initialize $V = 0$.
2. for $i = 1 : N$
3. $\alpha_i^{*} \leftarrow$ solution of LLC Eqn. (1.6)
4. $e_i = ||x_i - \Phi \alpha_i^{*}||_2$;
5. for $j = 1 : K$
6. for all locations $c$
7. $V(c) = V(c) + P(c - c_i, j)(1 - \delta(\alpha_j^{*}))e^{-\epsilon_i^2/\sigma^2}$
8. end
9. end
10. end

Table 4.1: Procedure of computing the final voting map to locate the target

as the coordinate origin (green dot in the right figure), the probability of the target location is represented by different red color level, with lighter color indicating a lower probability value.

We can see that the contribution from patches in region $a$ is smaller than those from region $b$, since their similarity to the basis are smaller than the ones in region $b$, as demonstrated by their coefficients. In this way, the probability map modeled in the dictionary can be used to compute the final voting map to find the target center through a spare representation.

Tracking stage: Denote $x_i$ as the $i$-th image patch centered at $c_i$ and $\alpha_j^{*}$ as its coefficients calculated by LLC. The overall target center voting map $V$ can be computed as:

$$V(c) = \sum_{i=1}^{N} \sum_{j=1}^{K} P(c - c_i, j)(1 - \delta(\alpha_j^{*}))e^{-\epsilon_i^2/\sigma^2},$$

(4.14)

where $\delta(x)$ is Dirac delta function. Only the probability map of those dictionary patches with non-zero coefficients will contribute to the final voting map. The $e^{-\epsilon_i^2/\sigma^2}$ in Eqn. (4.14) weights the voting by its sparse reconstruction accuracy. Patches with larger errors contribute less to the overall voting map. The details of the voting algorithm in the tracking stage are given in Algorithm 4.4.2. Using the voting map, the final tracking result can be found iteratively with

$$\hat{y} = \frac{\sum_{j=1}^{N'} c_j w_j V(c_j) g(||\frac{\hat{y}_0 - c_j}{h}||^2)}{\sum_{j=1}^{N'} w_j V(c_j) g(||\frac{\hat{y}_0 - c_j}{h}||^2)},$$

(4.15)
Algorithm 2: Tracking Framework Summary

**Input:** Target template

**Training Stage**
1. Extract target samples from a target template as training data $X$
2. Learn the target patch dictionary $\Phi$ using K-Selection in Sec. 3.2
3. Solve the sparse representation for each training data set by optimizing Eqn. (1.6)
4. Compute the target appearance model discussed in Sec. 4.3.1
5. Encode the target spatial information for each dictionary component using Eqn. (4.13)

**Tracking Stage**
1. Set the tracking result in the last frame as the initial guess of the target center in the current frame
2. For different scale changes in the range
3. Do image warping for a given scale and rotation to get $I'$
4. Extract the candidate samples from $I'$
5. Compute the coefficients for each sample using Eqn. (1.6)
6. Compute the voting map of target center with Eqn. (4.14)
7. Estimate the new target center with mean-shift iteratively
8. End
9. The tracking result with the highest score and corresponding scale and rotation will be considered as the final tracking result
10. Project the tracking result back to original image frame by doing an affine transformation with the tracked scale and rotation
11. Update the target model and voting map if online update enabled

Table 4.2: Summary of overall SPT tracking framework
Table 4.3: The challenges of the experimental sequences

<table>
<thead>
<tr>
<th>Sequence</th>
<th>3D Pose</th>
<th>Illumination</th>
<th>Occlusion</th>
<th>Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>√‡</td>
<td>√‡</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>girl</td>
<td>√‡</td>
<td>√</td>
<td>√‡</td>
<td>×</td>
</tr>
<tr>
<td>car</td>
<td>×</td>
<td>√‡</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>faceocc2</td>
<td>×</td>
<td>√‡</td>
<td>√‡</td>
<td>×</td>
</tr>
<tr>
<td>lemming</td>
<td>√</td>
<td>×</td>
<td>√‡</td>
<td>√</td>
</tr>
<tr>
<td>box</td>
<td>√</td>
<td>√</td>
<td>√‡</td>
<td>×</td>
</tr>
<tr>
<td>board</td>
<td>√‡</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>liquor</td>
<td>√</td>
<td>√</td>
<td>√‡</td>
<td>√</td>
</tr>
</tbody>
</table>

‡ High variation or occlusion.
† Partial illumination changes or occlusion.

4.5 Experimental Result

In this section, we evaluate our sparse tracking algorithm (SPT) on eight challenging sequences and compare its performance with five of the most recent existing state-of-the-art trackers. In addition, a set of experiments for celebrity face data and dynamic sequences with large amounts of motion were also tested to demonstrate the performance of SPT. For the comparison, either the binaries or source codes provided by the authors with the same initialization and parameter settings were used to generate the comparative results. The first three sequences (David, girl, car), the fourth sequence (faceocc2) and other sequences (lemming, box, board, liquor) can be downloaded from the URLs 1 2 3 respectively. The challenges for tracking presented by these sequences are summarized in Table 4.3, including pose variation, illumination changes, occlusion and scaling.

The discussion of parameter selection is presented in Section 4.5.1, followed by the comparative tracking results shown in Section 4.5.2. Overall, our proposed method provides the most accurate and stable results in these standard testing benchmarks for tracking.

1http://www.cs.toronto.edu/~dross/ivt/
2http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml
3http://gpu4vision.icg.tugraz.at/subsites/prost/prost.php
4.5.1 Parameter Analysis

Our algorithm has two important parameters: patch size $s$ and percentage $\beta$ of the selected basis over the whole training set. Four sequences (David, girl, faceocc2, board), exhibiting illumination changes, pose variations and occlusions were tested with $s = (3 \times 3, 5 \times 5, 7 \times 7, 9 \times 9, 11 \times 11)$ and $\beta = (5\%, 10\%, 15\%, 20\%, 25\%, 30\%)$.

As can observed in Figure 4.3(a), patch sizes 5 and 7 provide the best results. A dictionary learned with smaller image patches have more representational power, but less discriminative power. Similar trends can be observed for the percentage of selected basis vectors over the entire training set, illustrated in Figure 4.3(b). A larger dictionary will reduce tracking performance due to the loss of discrimination.

4.5.2 Comparative Tracking Results

In this section, the performance of our tracking algorithm was compared with other state-of-the-art methods.

Benchmark Sequences:

Our SPT method was first evaluated on four benchmark sequences compared with: multiple instance learning (MIL) [5], online simple tracking (PROST) [42], two stage sparse tracker (TST) [28] and incremental visual tracking (IVT) [41]. For the comparison to be fair, the mean ratio of a target center’s offset over the diagonal length of the target is used to measure the performance. The quantitative results are shown in Table 4.4. The best tracking results
Figure 4.4: Comparative results on benchmark sequences with our method (SPT), Multiple Instance Learning (MIL), Two Stage Tracker (TST) and Incremental Visual Tracker (IVT).

were emphasized in boldface. Our method produces the smallest tracking offset for the David, Girl, and Faceocc2 sequences which have the largest appearance variations. IVT yields slightly better results for the Car sequence which has high illumination changes but smaller appearance variations. MIL is good in general, but experiences difficulty in handling partial illumination changes, which may lead to drifting problems such as in the Car sequences. PROST generates relative stable results due to the combination of a static template and an online updated random forest.

The tracking results for selected frames from the David sequence are shown in Figure 4.5. The target face starts from a very dark environment to a much brighter one. The global illumination variations caused by the changing environment along the sequence can be easily handled by normalization. In contrast, the partial illumination and partial appearance changes do affect tracking accuracy for some of the other methods, such as frame #44, #296 for the MIL tracker.
Only selected features used in TST contribute to the good results under such difficult conditions. IVT generates very good results for this sequence because the appearance of the target face are linear at most of the frames, except for some frames like #159, #190. More detailed pixel-wise tracking errors (measured by the Euclidean distance from the center of the target to the ground-truth) are shown in Figure 4.4. The performance can also be compared in Figure 4.4(a). It is evident that our tracker generates very stable results with the smallest tracking error for this sequence.

Figure 4.6 presents the tracking results of comparisons between the different methods for the Girl sequence. The challenges of this sequence include 360 degree 3D pose variations and occlusions by another face which is similar to the target. Even though our method (using a static appearance dictionary) is not designed to handle large pose and appearance variations, it generates good results because small parts of the target can still produce a higher voting score to differentiate the target center from the background. The tracking offsets of our method become slightly larger for frames with quick pose variations (e.g. #314), because the encoded target configuration is not exactly matched. From Figure 4.4(b), we can see that MIL can generate relative stable results for this sequence through online updating, but it still cannot handle large and fast pose variations. IVT has more accurate results for those frames (e.g. #314) with rotations. However, the single, online updated subspace of IVT drifted to track another face as shown in frame #436. Our tracker generates very good results in occlusion scenarios such as in frame #436.

The car sequence was captured in an open road environment. The tracking results of the #32, #180, #200, #230, #240 and #352 are presented in Figure 4.7. The MIL method starts to show some target drifting (on the #200 frame) and finally loses the target (the #240 frame). IVT can track this sequence quite well. The target was successfully tracked using our proposed algorithm during the entire sequence. The TST method can accurately track the car also, but the bounding box is not as accurate as the one generated by our method and IVT. The detailed quantitative performance for this sequence is shown in Figure 4.4(c).

The face occlusion sequence is used to test the robustness of the proposed algorithm in handling occlusion. The #80, #150, #268, #540, #722 and #741 frames are presented in
Table 4.4: Comparative results on the benchmark datasets.

<table>
<thead>
<tr>
<th></th>
<th>david</th>
<th>girl</th>
<th>car</th>
<th>faceocc2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROST[42]</td>
<td>0.124</td>
<td>0.115</td>
<td>NA</td>
<td>0.116</td>
</tr>
<tr>
<td>TST[28]</td>
<td>0.052</td>
<td>0.131</td>
<td>0.065</td>
<td>0.139</td>
</tr>
<tr>
<td>MIL[5]</td>
<td>0.127</td>
<td>0.161</td>
<td>0.700</td>
<td>0.095</td>
</tr>
<tr>
<td>IVT[41]</td>
<td>0.059</td>
<td>0.147</td>
<td>0.020</td>
<td>0.081</td>
</tr>
<tr>
<td>SPT</td>
<td><strong>0.026</strong></td>
<td><strong>0.066</strong></td>
<td>0.031</td>
<td><strong>0.065</strong></td>
</tr>
</tbody>
</table>

Table 4.5: Comparative results on the PROST datasets.

<table>
<thead>
<tr>
<th></th>
<th>lemming</th>
<th>box</th>
<th>board</th>
<th>liquor</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROST[42]</td>
<td>0.189</td>
<td>0.091</td>
<td>0.157</td>
<td>0.101</td>
</tr>
<tr>
<td>FragTrack[1]</td>
<td>0.625</td>
<td>0.406</td>
<td>0.363</td>
<td>0.145</td>
</tr>
<tr>
<td>MIL[5]</td>
<td>0.112</td>
<td>0.740</td>
<td>0.206</td>
<td>0.619</td>
</tr>
<tr>
<td>SPT</td>
<td><strong>0.101</strong></td>
<td><strong>0.073</strong></td>
<td><strong>0.089</strong></td>
<td><strong>0.016</strong></td>
</tr>
</tbody>
</table>

Figure 4.8. The MIL algorithm can roughly capture the position of the object, but did show some drifting problems when there is heavy occlusion, shown in the #741 frame. IVT starts to fail after the #540 frame. Our proposed method and TST provide reasonable results. Similarly to the car sequence, SPT still provides more accurate bounding boxes than TST. The detailed quantitative performance for this sequence is shown in Figure 4.4(d).

**PROST Sequences:** To further evaluate our SPT method for accurate tracking under occlusion, appearance blur, and pose variation, the latest four sequences provided in [42] are selected for comparison with PROST, MIL, and FragTracker [1]. Our SPT method produced the best performance for all sequences, as shown in Table 4.5, while PROST yielded the second best performance. Pixel-wise tracking results and the results of selected frames show that the other methods have difficulties in accurately locating the target under heavy occlusion, as we show in the #336 frame in the lemming sequence (Figure 4.9), the #300 in the box sequence (Figure 4.10), and in the #731 frame in the liquor sequence (Figure 4.12(d)). MIL and PROST cannot track the target accurately when large pose variations occurs, as we show in the #994 frame in the lemming sequence (Figure 4.9), the #600 frame in the box sequence (Figure 4.10), the #497 frame in the board sequence (Figure 4.12(b)), while our SPT method can track the target even under 90 degree off-plane rotation, as we shown in the #497 frame in the board sequence (Figure 4.12(a)) and the #731 frame in the liquor sequence (Figure 4.12(c)).

**Celebrity Face Data:** In Figure 4.13, tracking results for selected frames from a celebrity face dataset in [36] are presented. There is much variation of expression and 3D pose, as well
as scale changes in the sequence. As can be seen from the results, our proposed method can generate very accurate tracking results, even when only half of the face is visible.

**Dynamic Motion Sequences:** To demonstrate that our method works in a dynamic scenario, the proposed algorithm is tested on three additional sequences (animal, football, and basketball), illustrated in Figure 4.14. The targets in all three videos have large motions. The top row in Figure 4.14 represents the tracking result of an animal sequence. The target was running on the water with other animals exhibiting a similar appearance. As can be seen from the experimental results, our tracker can generate accurate results even with heavy motion blur and water occlusion. The middle row in Figure 4.14 presents the tracking results of a football game sequence with a large amount of similar background clutter. It can be observed that our proposed tracking method can follow the football player very well. The basketball sequences shown in the bottom row demonstrated that our patch-based voting methods can handle non-rigid targets with a reasonable dynamic spatial configuration. However, a large, non-rigid transformation might fail the proposed tracking method, since the learned target center configuration relative to each part will not hold any more. An adaptive tracking approach using an online update voting map will be discussed in Section 4.6.

4.6 Discussion

In this section, I discuss potential shortcomings of the SPT method, and possible ways of overcoming them, together with future work.

**Potential Shortcomings of the SPT Method** In this paper, I have proposed a general tracking framework to integrate both the appearance model and spatial configuration of the target. However, only the relative target center information is encoded into the dictionary and then retrieved for locating the target along the sequence. There are some potential cases where the method may fail due to this constraint.

- Large movement between consecutive frames might lead to potential failure of SPT since the target center is found by iteratively moving from the initial guess to the final solution. However, relatively small perturbations between consecutive frames will not present a problem.
• Unseen appearance changes will reduce the validity of the learned appearance model and lead to possible errors. To balance stability and flexibility, the dictionary in SPT is kept static along the video sequence after the first frame is learned. When the target has large appearance changes this may lead to tracking failure.

• Targets with uniformly distributed appearance patches, such the T-shirt with a single color, can lead potentially to tracking failures. In SPT, the spatial configuration and basis distribution are used to locate the target center. This type of target will generate a flat voting map and spike histogram, and thus reduce the robustness of SPT. A way to alleviate this potential shortcoming is to use a larger kernel, so that the patches around the target boundary can provide a sufficient contribution to locate the target center.

*Scaling:* Mean-shift and voting methods can locate the target center very accurately, but they cannot handle scaling in natural scenarios. Without handling scaling, the appearance of patches will not match the learned target appearance model. In our method, we assume that the target’s scaling is continuous, and therefore keep recording the target scale from previous frames and try different scales in the current frame. The scale with the best tracking result (highest score) will be selected as the final target-object scale.

*Rotation:* Large rotations or non-rigid transformations will make the learned sparse appearance model and learned voting map invalid, thus generating inaccurate results as shown in the left column of Figure 4.15. To provide better performance for rotation, there are several methods that can be used and will be pursued in future work. The first one is to perform tracking in the affine transformed image frame for different rotation parameters similar to the way we handle the scaling in SPT. The tracking result with the highest score will be considered to be the final result. It is based on the assumption that the rotation is continuous. However, this method is not efficient since the tracking must be performed multiple times for different rotation parameters.

To add more flexibility to the voting map, an online updated voting map can be concatenated to the original voting map learned in the first frame. In this way, the voting map can adapt to the rotation. The right column of Figure 4.15 denotes the tracking result with an online updated voting map. It has been already shown that the combination of a static and a dynamic updated
voting map can generate more robust tracking results, especially for rotation.

As future work for handling rotation and scaling, it is worth exploring the dictionary learning pyramid with various scalings and rotations in the training stage. Affine transformation states can be encoded and retrieved for tracking the target in the same manner as the sparse voting algorithm.
Figure 4.5: Result of selected frames from the David sequence with our method (SPT), Multiple Instance Learning (MIL), Two Stage Tracker (TST) and Incremental Visual Tracker (IVT).

<table>
<thead>
<tr>
<th>SPT</th>
<th>MIL</th>
<th>TST</th>
<th>IVT</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Frame #44" /></td>
<td><img src="image2" alt="Frame #159" /></td>
<td><img src="image3" alt="Frame #190" /></td>
<td><img src="image4" alt="Frame #296" /></td>
</tr>
<tr>
<td><img src="image5" alt="Frame #390" /></td>
<td><img src="image6" alt="Frame #390" /></td>
<td><img src="image7" alt="Frame #390" /></td>
<td><img src="image8" alt="Frame #390" /></td>
</tr>
<tr>
<td><img src="image9" alt="Frame #452" /></td>
<td><img src="image10" alt="Frame #452" /></td>
<td><img src="image11" alt="Frame #452" /></td>
<td><img src="image12" alt="Frame #452" /></td>
</tr>
</tbody>
</table>
Figure 4.6: Result of selected frames from the Girl sequence with our method (SPT), Multiple Instance Learning (MIL), Two Stage Tracker (TST) and Incremental Visual Tracker (IVT).
Figure 4.7: Result of selected frames from the Car sequence with our method (SPT), Multiple Instance Learning (MIL), Two Stage Tracker (TST) and Incremental Visual Tracker (IVT).
Figure 4.8: Result of selected frames from the Faceocc2 sequence with our method (SPT), Multiple Instance Learning (MIL), Two Stage Tracker (TST) and Incremental Visual Tracker (IVT).
Figure 4.9: Result of selected frames from the Lemming sequence with our method (SPT), Simple Tracker (PROST), Multiple Instance Learning (MIL) and Fragment based Tracker (FragTrack).
Figure 4.10: Result of selected frames from the Box sequence with our method (SPT), Simple Tracker (PROST), Multiple Instance Learning (MIL) and Fragment based Tracker (FragTrack).
Figure 4.11: Comparative results on the PROST sequences with our method (SPT), Multiple Instance Learning (MIL), Simple Tracker (PROST) and Fragment based Tracker (FragTrack).
Figure 4.12: Result of selected frames from our method (SPT), Multiple Instance Learning (MIL), Simple Tracker (PROST) and Fragment based Tracker (FragTrack).
Figure 4.13: Tracking result of celebrity face data in [36]. The tracking results are labeled as red rectangle.

Figure 4.14: Tracking experiments on sequences with more motion dynamics
Figure 4.15: The inaccurate tracking results with a static voting map (left column), and those corrected with an online updated voting map (right column)
Chapter 5  
Motion Registration Guided Online Collaborative Trackers  

5.1 Introduction  
Accurate tracking of tumor movement in fluoroscopic video sequences is a clinically significant and challenging problem. It has been well demonstrated that tumors located in the thorax may exhibit significant respiratory induced motions[16]. These physiologically related motions may increase the target positioning uncertainty and have a negative impact on radiation treatment.

Recent computer-aided methods for tracking the tumor in fluoroscopic videos can be categorized into three groups: (1) Finding the tumor positions based on the external surrogates[23]; (2) Tumor tracking with the help of fiducial markers inside or near the tumor [9]; (3) Tumor tracking without implanted fiducial markers. The optical flow [51] produces promising tracking results when there is relatively small motion and clear boundaries between adjacent frames. Shape models of individual annotated tumors at different phases of respiration were learned offline to achieve good tracking results [52]. A motion model and one-step-forward prediction were applied to reliably track the left ventricle in 3D ultrasound [54]. However, these methods require a lot of expensive annotations, and usually cannot produce satisfactory results when the tumor boundary is not clear.

In Figure 5.1 we show one typical fluoroscopic image, with the position of the tumor and the contour of the lung overlaid on the right subfigure. From the image we can tell there are two major challenges for tracking in fluoroscopic videos: 1) Due to the contrast and possible occlusion from the heart, the boundary of the tumor is not clear. Actually for initialization, the doctor applied the 4DCT to help the annotation on the fluoroscopic image. 2) Due to
the unclear boundary of the tumor, it is extremely labor intensive for the human experts to provide annotation for each frame in the whole sequence. Therefore many tracking algorithms based on offline learning or local features might not be accurate without being provided an accurate prior. The major contribution of this work is how we address both challenges: 1) we propose to provide the tumor prior by registering the lung, or part of the lung, which has clear boundary and can be easily tracked; 2) In order to avoid the requirement for a large amount of annotation, we propose to develop an online tracking algorithm to refine the boundary of the tumors incrementally. Section 2 introduces the online contour tracking framework. The lung tracking, tumor registration and online collaborative tracking algorithm are described in Section 3 and Section 4. Section 5 provides the experimental results and Section 6 concludes the paper.

5.2 Online Contour Tracking Framework

The tumor contour is represented as a list of landmark points in clockwise order. We denote the contour as \( C = \{c_1, c_2, ..., c_n\} \), where \( c_i \) is the \( i \)-th landmark point. Let \( Z = \{z_1, z_2, ..., z_n\} \) as the observation of the contour, \( \Lambda = \{x_1, x_2, ..., x_n\} \) as the states of the contour, and \( x_i = \{x, y\} \) as the coordinates of landmark point \( c_i \). The tracking problem can be represented as the estimation of a state probability \( p(\Lambda_t|Z_{1:t}) \), where \( Z_t \) is the observation at the \( t \)-th frame.

\[
p(\Lambda_t|Z_{1:t}) \propto p(Z_t|\Lambda_t)p(\Lambda_{t-1}|Z_{1:t-1})
\] (5.1)
\[ \Lambda_t^* = \text{argmax}_{\Lambda_t} p(\Lambda_t | Z_{1:t}) \] is the estimation of the contour tracking result.

Assuming each landmark point is independent to each other, let \( x_t \) and \( z_t \) serve as the state and observation of the \( i \)-th landmark point in the \( t \)-th frame. Then (5.1) is equivalent to:

\[
p(\Lambda_t | Z_{1:t}) \propto \prod_{i=1}^{n} p(x_t | z_{t_1}, z_{t_2}, \ldots, z_{t_t}). \tag{5.2}
\]

Tracking results for all landmark points with a maximum \( p(x_t | z_t) \) yield the optimal solution of (5.2). For convenience, we present the contour tracking as the estimation of \( p(x_t | z_t) \) for each point. Bayesian importance sampling estimates and propagates the probability by recursively performing prediction:

\[
p(x_t | z_{1:t-1}) = \int p(x_t | x_{t-1}) p(x_{t-1} | z_{1:t-1}) dx_{t-1} \tag{5.3}
\]

and updating

\[
p(x_t | z_{1:t}) \propto p(z_t | x_t) p(x_t | z_{1:t-1}). \tag{5.4}
\]

where \( p(x_t | z_{1:t-1}) \) represents the prior and \( p(x_t | z_{1:t}) \) is the likelihood.

### 5.3 Tumor Contour Prediction Using Lung Motion Registration

One of the key ideas in our tumor tracking algorithm is based on the following two facts: 1) *Although the boundary of the tumor is not easily detected, there are some parts of the lung with higher contrast and a clearer boundary which can be easily tracked;* 2) *The tumor never moves itself, the motion of the tumor can be roughly inferred from the motion of the lung.* Based on these two facts, by tracking the motion of the lung between consecutive frames, we can calculate the nonlinear transformation field between two adjacent frames. Therefore the initial position of the tumor position in the next frame can be inferred from the inter-frame motion of
Figure 5.2: The computation of the motion prior of the tumor using nonlinear registration of the lung between consecutive frames.

In equation (5.3), the transition model \( p(x_t | x_{t-1}) \) is usually constrained by assuming a Gaussian distribution \( N(x_t | x_{t-1}, \sigma) \). However, we argue that for real clinical cases the motion prior of the tumor may not follow a Gaussian distribution. In our algorithm, we propose to model the motion prior \( p(x_t | x_{t-1}) \) by registering the accurately tracked lung motion between adjacent frames and applied the calculated nonlinear transformation to predict the motion of the tumor in the next frame.

In our algorithm, we calculate the motion prior (state model) by motion registration, which is illustrated in Figure 5.2. At time \( t \), we first compute the tracking results of the lung \( w \) using simple template matching. Given \( w \) as a 2D contour point and its neighborhood \( N(w) \), let \( G(N(w), \mu) \) denotes the transformation of the template. (The neighborhood was chosen to be a \( 20 \times 20 \) window) The goal is to search the best transformation parameters which minimize the error between \( N(w) \) and \( G(N(w), \mu) \).

\[
\mu = \arg \min_{\mu} \sum_{w \in N(w)} [G(N(w), \mu) - N(w)]^2 .
\] (5.5)

Record the contour of the lung in time \( t-1 \) as \( b \), we will be able to calculate the nonlinear transformation, e. g. \( T1 \) in Figure 5.2, which maps \( b \to w \). In our algorithm thin plate
spline (TPS) transformation [7] is applied to perform this mapping. The TPS is a nonrigid transformation between two 2D point sets. The transformation $T$ contains an affine mapping plus a wrapping coefficient matrix. We used 150 uniformly sampled points on the contour of the lung to estimate the nonlinear transformation $T$ by minimizing

$$E_{TPS}(T) = \sum_{i=1}^{150} \|w_i - T(b_i)\|^2 + \lambda f(T) \quad (5.6)$$

where $w_i$ denote the 2D lung boundary point at time $t$ and $b_i$ denote the points on the lung boundary at time $t - 1$. The $f(T)$ is a function containing a kernel which represents the internal structure relationship of the point set. The regularization parameter $\lambda$ is chosen as 1.5. After we calculated the nonlinear mapping $T$, the final prediction result of the tumor contour $p(x_t)$ is calculated using $T(p(x_{t-1}))$. This step is iterated until the end of the sequence, as shown as $T1$, $T2$, etc. in Figure 5.2. The final tumor tracking results are refined using online collaborative trackers.

### 5.4 Online Collaborative Tracking

As current offline learning based tracking approaches require a large amount of manual labeling, which is impractical in our case, we present an online collaborative tracking algorithm of lung tumors in fluoroscopic video sequences. Adaptive classifiers are incrementally "taught" to discriminate landmark points in the contour from others. In equation (5.4), The observation model $p(z_t|x_t)$ represents the likelihood of $z_t$ being generated from state $x_t$. In our algorithm, it is measured with two independent online learned models collaboratively. One is a discriminative model using an asymmetric online updated classifier. The other is based on a generative model, which represents the appearance of each small region around the landmark point in an incrementally learned low dimensional subspace. For point observation $z_t$ in the current frame, the observation model is $p(z_t|x_t) \propto p_D(z_t|x_t)p_G(z_t|x_t)$, where $p_D(z_t|x_t)$ and $p_G(z_t|x_t)$ are the likelihood calculated from the discriminative and generative models, respectively. The final contour tracking result $x_t = \arg\max_{x_t} p(x_t|z_{1:t})$. 
Algorithm: Asymmetric online boosting

Define: Let \( \{x_i, y_i\}_{i=1}^{N} \) as the dataset, \( x_i = \{x_{i1}, ..., x_{id}\} \) is feature vector and \( y_i \in \{-1, 1\} \) is its label. The \( \gamma \) is the learning rate. \( \omega = \{\omega_1, \omega_2, ..., \omega_N\} \) is the samples’ weight. The \( H_t(x) \) is the strong classifier in the \( t \)-th iteration with confidence function \( f_t(x) \), the weighted accuracy and error are denoted as \( \lambda_{\text{cor}}^m \) and \( \lambda_{\text{wro}}^m \), \( m = 1, ..., M \), where \( M \) is number of weak classifiers. The \( I \) is the indicator function.

1. \( \omega_i = 1, i = 1, ..., N \)
2. for \( t = 1, 2, ..., T \) do
   3. \( \omega_i = \omega_i \exp(y_i \sqrt{k}/T) \), \( \omega_i = \omega_i / \sum_{j=1}^{N} \omega_j \)
   4. for \( m = 1, 2, ..., M \) do
      5. \( h_m(x) = \text{update}(h_m(x), \{x_i, y_i\}_{i=1}^{N}, \omega) \)
      6. \( \lambda_{\text{cor}}^m = \lambda_{\text{cor}}^m (1 - \gamma) + \gamma \sum_{i=1}^{N} \omega_i * I(h_m(x_i) = y_i) \)
      7. \( \lambda_{\text{wro}}^m = \lambda_{\text{wro}}^m (1 - \gamma) + \gamma \sum_{i=1}^{N} \omega_i * I(h_m(x_i) \neq y_i) \), \( \epsilon_m = \lambda_{\text{wro}}^m / (\lambda_{\text{cor}}^m + \lambda_{\text{wro}}^m) \)
   8. end for
   9. \( m^* = \arg\min_m \epsilon_m \), \( \alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_{m^*}}{\epsilon_{m^*}} \right) \), \( \omega_i = \omega_i \exp(-y_i * h_{m^*}(x_i)) \)
   10. \( f_t(x) = f_{t-1}(x) + \alpha_t h_{m^*}(x) \), \( H_t(x) = \text{sign}(f_t(x)) \)
11. end for

5.4.1 Asymmetric Online Boosting

We define training samples \( \{x_i\}_{i=1}^{N} \in \mathbb{R}^d \) and their labels \( \{y_i\}_{i=1}^{N} \in \{-1, 1\} \), where \( y = 1 \) denotes a landmark point and \( y = -1 \) is a background point. A function \( f(x) : \mathbb{R}^d \rightarrow \mathbb{R} \) can be learned as the confidence of labeling sample \( x \). \( x \) is classified to be the object if \( f(x) > 0 \) or background otherwise. The \( p_{D}(z_t|x_t) \) is measured by \( f(x) \), where \( x \) is the sample generated with state \( x_t \). Because the training samples are unbalanced and provided incrementally, the discriminative model is trained using asymmetric online boosting. This method provides a more robust classifier and converges faster for unbalanced training sets [47].

The asymmetric online boosting method used in our algorithm is summarized in Algorithm 5.4.1. The strong classifier \( H(x) \) is updated incrementally using tracking results in the current frame. Each weak learner \( h_i(x) \) and its corresponding weight \( \alpha_i \) are updated with learning rate \( \gamma \) (shown in Algorithm 5.4.1). This online updating schema enabled the incrementally trained classifier to be more adaptive to gradual appearance changes.
5.4.2 Dynamic Appearance Model and Subspace Learning

Define $I = (I_1, I_2, ..., I_n)$ as $n$ observations of one landmark point in $n$ consecutive frames, $I$ can be modeled as a space spanned by low dimensional orthogonal subspace $U_{q \times k}$ centered at $\mu$, where $q$ is the length of the vectorized model and $k$ is the dimension of the subspace. The $U$ is the low dimension eigenvectors of the samples’ covariance matrix $\frac{1}{n} \sum_{i=1}^{n} (I_i - \mu)(I_i - \mu)^T$, and $\mu = \frac{1}{n} \sum_{i=1}^{n} I_i$ is the moving average.

Let $\Omega = \{\mu, U, \Sigma\}$ serves as the appearance model for $I$, $\Sigma$ is a diagonal matrix contains $k$ largest singular values of $I$ and $\lambda_1 > \lambda_2 > ... > \lambda_k$. For a given sample $I_{\chi_t}$ extracted in state $\chi_t$, the likelihood of $I_{\chi_t}$ generated from $\Omega$ can be decomposed into the probabilities of distance-within-subspace \[ p_U(I_{\chi_t}|\Omega) \propto \exp\left(-\frac{1}{2}(I_{\chi_t} - \mu)^T U \Sigma^{-2} U^T (I_{\chi_t} - \mu)\right) \tag{5.7} \]
and distance-to-subspace,

\[ p_{\bar{U}}(I_{\chi_t}|\Omega) \propto \exp\left(-||I_{\chi_t} - \mu - UU^T (I_{\chi_t} - \mu)||^2\right). \tag{5.8} \]

Let $A$ denote the observations in the previous $n$ frames and $B$ represent the most recent $m$ frames. Incremental subspace learning [41] is performed to merge the new frames into the original subspace learned from $A$. With $A = U \Sigma V^T$ by singular value decomposition (SVD), $\hat{B}$ as the concatenated centered data of $B$ plus one additional vector $\sqrt{\frac{nm}{n+m}} (\mu_B - \mu)$, and $\tilde{B}$ as the components of $\hat{B}$ orthogonal to $U$, the concatenated matrix of $A$ and $B$ then can be written as

\[ [A \ B] = [U \ \hat{B}] R \begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix}, \tag{5.9} \]
where $R = \begin{bmatrix} \Sigma & U^T \hat{B} \\ 0 & \hat{B}(\hat{B} - UU^T \hat{B}) \end{bmatrix}$.

After we calculate the SVD of $R = \tilde{U} \Sigma \tilde{V}^T$, the new subspace is updated as $U' = [U \ \tilde{B}] \tilde{U}$.
and $\Sigma' = \tilde{\Sigma}$. The mean $\mu' = \frac{n\gamma}{n\gamma + m} \mu_A + \frac{m}{n\gamma + m} \mu_B$ is updated with a forgetting factor $\gamma$ to decrease the weight of the older appearance models.

5.5 Experimental Results

Twelve sets of fluoroscopic video sequences were collected to test our algorithm. Eight sets were from four different patients and the other four sets were from one patient. The fluoroscopic images were acquired for lung cancer patients who were undergoing radiotherapy. These fluoroscopic images were saved in digital format and readily available for video display and analysis. Each sequence lasts for about 10 seconds and covers three to five respiration cycles. We have both posterior-anterior (PA) fluoroscopic video sequence, and the lateral sequence. As discussed previously, the quantified moving information is extremely important in radiotherapy management of lung cancers. Using the prescribed algorithm, an optimal plan and treatment strategy can be designed to provide a desired conformal dose coverage to a tumor target while sparing as much surrounding normal tissues as possible.

The experiments of the developed algorithm were conducted as follows: The initial contours for these sequences were labeled in a 3D CT by an experienced radiation oncologist. On the 3D CT images, the target (cancer tumor) could be clearly identified and delineated. Digitally reconstructed radiograph, along with the delineated contours, were projected along the PA and lateral directions. Based on the digitally reconstructed radiograph and projected target, an experienced radiation oncologist manually identified and delineated the target. The comparative tracking results of four sequences are shown in Figure 5.3. The result of proposed tracker are presented in top row, the result of collaborative tracker without prior and optical flow are presented in the middle and bottom rows. The errors are marked with white rectangle. Sequence 1 contains 175 frames and the result of frame (58, 98, 224) are displayed in Figure 5.3(a). Sequence 2 contains 60 frames and the result of frame (124, 172, 180) are displayed in (b). The result of Sequence 3 and 4 contains 75 and 56 frames with result shown in (c) and (d). The proposed method performs well even when the tumor boundary is not clear.

The human experts have annotated roughly 10-20 frames for each sequence. Using these ground truth contours, we measure the quantitative tracking accuracy based on point-to-contour
Figure 5.3: The comparative tracking results of four sequences (a-d) with proposed method (top row), online collaborative tracking (middle row) and optical flow (bottom row).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean</th>
<th>Variance</th>
<th>Median</th>
<th>Min</th>
<th>Max</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D optical flow tracking</td>
<td>2.75</td>
<td>1.69</td>
<td>2.34</td>
<td>0.70</td>
<td>6.70</td>
<td>4.22</td>
</tr>
<tr>
<td>Online tracking without prior</td>
<td>2.49</td>
<td>1.36</td>
<td>2.58</td>
<td>0.59</td>
<td>5.58</td>
<td>3.70</td>
</tr>
<tr>
<td>Registration guided online tracking</td>
<td>1.75</td>
<td>0.81</td>
<td>1.70</td>
<td>0.59</td>
<td>3.72</td>
<td>2.36</td>
</tr>
</tbody>
</table>

(PTC) error. All 2D points on each frame of the testing sequence are projected onto the corresponding annotated boundary of the test set. The projection distance from the point to boundary is recorded as the PTC error, $e_{ptc}$. For a perfect tracking, the $e_{ptc}$ should be equal to zero for each 2D frame. In Table 5.1, we compared the quantitative $e_{ptc}$ using our proposed algorithm, with tracking by optical flow and tracking by collaborative trackers without prior. The 80% column in Table 5.1 represents the sorted 80% smallest error of all $e_{ptc}$. The mean $e_{ptc}$ we obtained is 1.75 pixel with a 80% error below 2.36 pixels.
5.6 Conclusion

We have presented motion registration guided online collaborative trackers for lung tumor tracking in fluoroscopic video sequences. This method addresses the challenges of landmark ambiguity along the boundary of the tumor by tracking the clear boundary of the lung first, then inferring the contour of the tumors. Because the whole tracking framework is based on online learning, our algorithm does not require expensive expert annotations and therefore is a cost saving approach. The experimental results demonstrate the effectiveness of our method. Our algorithm is quite general and can be extended to other medical object tracking applications: especially for those who do not have clear boundaries, and/or for which large amounts of annotations are difficult or expensive to obtain.
Chapter 6
Conclusions and Future Directions

In this dissertation, we investigate a set of algorithms for sparse representation in visual tracking. We have presented a novel dictionary learning algorithm for sparse representation named K-Selection. This algorithm selects the most representative data from the training set to be the dictionary components. The selected dictionary has preferred feature desired in visual tracking application in which differentiating the target from background is more important than sparsely representing the target.

With the dictionary learning algorithm, we also proposed a novel tracking algorithm using local appearance model. The target appearance is modeled by its local patches, which can be more general for covering dynamic appearance changes. This local appearance model using image patches is also more robust to occlusions. A novel sparse representation based voting map is proposed to locate the target efficiently. The advantages of this tracking algorithm were demonstrated with a set of comprehensive experiments.

Besides the above main contributions, I have also presented the tracking algorithm using two-stage sparse optimization. The two-stage sparse optimization emphasize both the sparsity of templates necessary for sparse representation and the sparsity of features necessary for differentiate the target from the background clutters. At the end of the dissertation, I also discussed an application of tracking in medical image analysis.

Many concepts and algorithms presented in this dissertation are general and can be extended to other computer applications.
References


