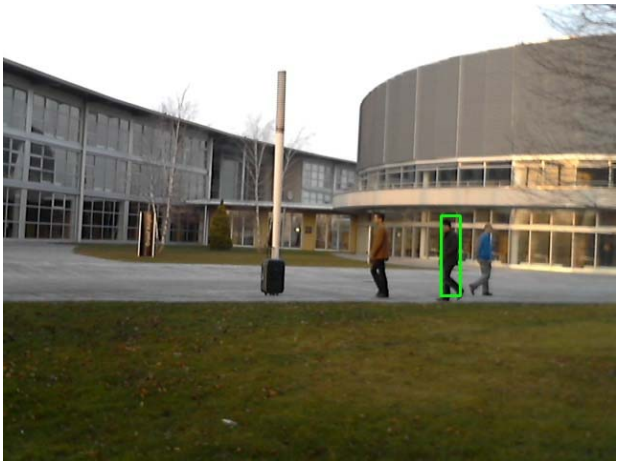
Online machine learning methods for visual tracking

Lei Qin

Thèse
de doctorat
de l'UTT

Lei QIN

Online Machine Learning Methods for Visual Tracking



Spécialité :
Optimisation et Sûreté des Systèmes

2014TROY0017 **Année 2014**

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Lei QIN

le 5 mai 2014

Online machine learning methods for visual tracking

JURY

M. I. NIKIFOROV	PROFESSEUR DES UNIVERSITES	Président
M. F. ABDALLAH	MAITRE DE CONFERENCES - HDR	Directeur de thèse
M. F. DORNAIKA	PROFESSOR	Rapporteur
M. T. RODET	PROFESSEUR DES UNIVERISTES	Rapporteur
M. M. SAHMOUDI	ENSEIGNANT CHERCHEUR ISAE TOULOUSE	Examinateur
M. H. SNOUSSI	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors, Prof. Hichem SNOUSSI and Dr. Fahed ABDALLAH, for all of their help, support and guidance throughout the duration of my thesis in the past four years. During my research stay at the Université de Technologie de Troyes, I have benefited enormously from their valuable comments, their innovative approaches to research, and their persistent pursuit of achieving results of high quality.

A special thank goes to Prof. Thomas RODET (ENS, Cachan), Prof. Fadi DORNAIKA (University of the Basque Country), and Prof. Igor NIKIFOROV (Université de Technologie de Troyes), Dr. Mohamed SAHMOUDI (Institut Supérieur de l'Aéronautique et de l'Espace, Campus SUPAERO), who kindly agreed to serve as reviewer (rapporteur) or examiner (examinateur) in my Ph.D defense committee.

I am grateful to all secretaries who had helped me: Ms. Ling GONG of the international office of UTT, Madame Veronique BANSE, Marie-jose ROUSSELET and Bernadette ANDRÉ in the department of Optimization and Security of Systems (OSS), Madame Isabelle LECLERCQ, Pascale DENIS and Therese KAZARIAN in the doctoral school of UTT. Thanks for their kindness and friendliness during my study at UTT. At the same time, I want to also thank all colleagues in the laboratory for sharing the research environment and experience.

I would like to express my deep gratitude to my family. During the entire research process, my parents provided me with unconditional support and encouragement. I could always rely on them and they gave me the strength I needed to achieve this goal.

The funding for my PhD study was provided by the China Scholarship Council.

Lei QIN

Résumé

Nous étudions le problème de suivi de cible dans une séquence vidéo sans aucune connaissance préalable autre qu'une référence annotée dans la première image. Pour résoudre ce problème, nous proposons une nouvelle méthode de suivi temps-réel se basant sur à la fois une représentation originale de l'objet à suivre (descripteur) et sur un algorithme adaptatif capable de suivre la cible même dans les conditions les plus difficiles comme le cas où la cible disparaît et réapparait dans le scène (ré-identification). Tout d'abord, pour la représentation d'une région de l'image à suivre dans le temps, nous proposons des améliorations au descripteur de covariance. Ce nouveau descripteur est capable d'extraire des caractéristiques spécifiques à la cible, tout en ayant la capacité à s'adapter aux variations de l'apparence de la cible. Ensuite, l'étape algorithmique consiste à mettre en cascade des modèles génératifs et des modèles discriminatoires afin d'exploiter conjointement leurs capacités à distinguer la cible des autres objets présents dans la scène. Les modèles génératifs sont déployés dans les premières couches afin d'éliminer les candidats les plus faciles alors que les modèles discriminatoires sont déployés dans les couches suivantes afin de distinguer la cibles des autres objets qui lui sont très similaires. L'analyse discriminante des moindres carrés partiels (AD-MCP) est employée pour la construction des modèles discriminatoires. Enfin, un nouvel algorithme d'apprentissage en ligne AD-MCP a été proposé pour la mise à jour incrémentale des modèles discriminatoires.

Mots clés: apprentissage automatique, analyse multivarié, analyse de covariance, détection du signal.

Abstract

We study the challenging problem of tracking an arbitrary object in video sequences with no prior knowledge other than a template annotated in the first frame. To tackle this problem, we build a robust tracking system consisting of the following components. First, for image region representation, we propose some improvements to the region covariance descriptor. Characteristics of a specific object are taken into consideration, before constructing the covariance descriptor. Second, for building the object appearance model, we propose to combine the merits of both generative models and discriminative models by organizing them in a detection cascade. Specifically, generative models are deployed in the early layers for eliminating most easy candidates whereas discriminative models are in the later layers for distinguishing the object from a few similar "distracters". The Partial Least Squares Discriminant Analysis (PLS-DA) is employed for building the discriminative object appearance models. Third, for updating the generative models, we propose a weakly-supervised model updating method, which is based on cluster analysis using the mean-shift gradient density estimation procedure. Fourth, a novel online PLS-DA learning algorithm is developed for incrementally updating the discriminative models. The final tracking system that integrates all these building blocks exhibits good robustness for most challenges in visual tracking. Comparing results conducted in challenging video sequences showed that the proposed tracking system performs favorably with respect to a number of state-of-the-art methods.

Key words: machine learning, multivariate analysis, analysis of covariance, signal detection.

Contents

List of Figures

List of Tables

To my parents.

Chapter 1

Introduction

1.1 The visual object tracking problem

Visual object tracking is a fundamental task within the field of computer vision. The use of visual tracking is pertinent in a variety of applications, such as automated surveillance, video indexing, human-computer interaction, vehicle navigation, and augmented reality.

In general, tracking is a challenging task which consists in generating an inference about the motion of an object given a sequence of images. In this thesis, we consider a simplified definition, which has been adopted by a majority of works in literature: starting from a bounding box given in the first frame, tracking is the image analysis problem for the purpose of localizing an arbitrary object over a sequence of frames [7]. In other words, we focus on the problem of tracking an arbitrary object in a video sequence with no prior knowledge other than the location of the object in the first frame. An illustrative sample of visual tracking is presented in Figure 1.1.

FIGURE 1.1: An illustrative example of visual tracking. Left: the target annotated in the first frame. Right: the tracked result in green rectangle and the tracking trajectory in blue.

The challenges arising in visual tracking are due to a number of factors: the loss of information caused by the projection of the 3D world on a 2D image, noise in images, cluttered-background, complex object motion, partial or full occlusions, illumination changes etc. In long-term unconstrained environments, this problem is even more challenging as the target can be fully occluded or can leaves the field of view for a long time and requires the tracker to re-acquire it and to continue the tracking when it re-appears in the visual scene.

To build a robust tracking system, some requirements should be considered. First, the tracking algorithms need to be able to follow the object of interest even under complicated conditions. Second, additional to various changes of the environment, the object itself may also undergo appearance changes. This requires a steady adaptation mechanism of the tracking system to the actual object appearance. Third, for practical use, the speed of the system is of great concern for real-time implementation. Therefore, selection of fast algorithms as well as optimized implementation are required.

1.2 Components of a typical visual tracking system

To tackle the visual tracking problem defined above, the majority of successful tracking systems in literature are appearance-based methods. In these methods, image regions are represented by their feature descriptors describing their appearances.

For localizing the target, the object has an appearance model derived from the feature representation and a metric-based cost function associated with the model. This cost function evaluates the similarity, likelihood, distance or other quantities alike of a candidate descriptor to the object appearance model. Localization of the target is then achieved by finding a candidate region that optimizes the cost function. For tracking purposes, a motion model is also needed for predicting possible locations of the target, i.e. candidate regions, based on the current target position and a prior spatial motion model. After localization, an adaptive tracker usually takes into account the observation in the current frame and updates the object appearance model to adjust to potential changes of the target or the background.

We depict in Figure 1.2 a reference model structuring most of the appearance-based tracking systems. This reference model decomposes a system into four components, namely motion representation, appearance description, object appearance model with its decision metric, and updating of the object appearance model. Note that this reference model can be simplified. For instance, some appearance-based methods may not include a model updating mechanism, only a fixed static appearance model is used.

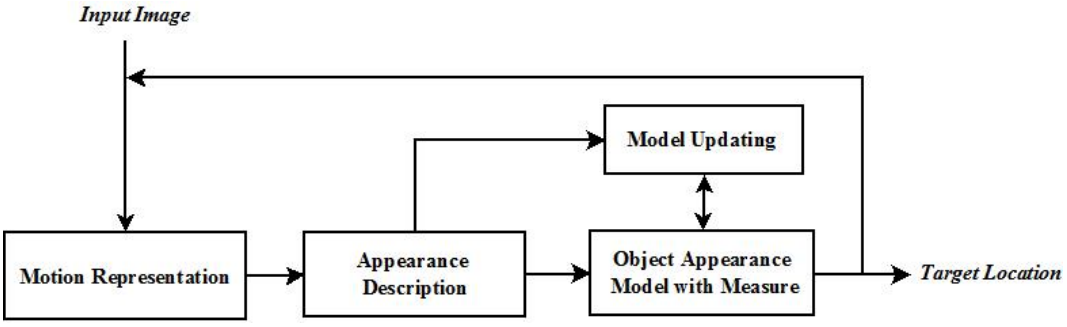


FIGURE 1.2: Diagram of a typical appearance-based tracking system. When a new frame is captured, the tracking process works as follows. First, the motion model proposes a set of possible states of the target. The feature description component then extract features or descriptor to represent each of the candidates. The localization of the target is accomplished by finding the candidate that optimizes the quantity measure associated with the object appearance model. For adaptive systems, the object appearance model is then updated using the new observations obtained in this frame. Besides, the estimated target location may be utilized by the motion model for proposing candidate regions in the next frame.

1.3 Main contributions

In this thesis, we propose novel approaches for visual tracking. More specifically, our contributions concern three of the four components of an appearance-based tracker, except the motion representation component, namely the feature extraction, the feature descriptor updating and the decision algorithm. The effectiveness of these approaches are assessed empirically and/or theoretically.

For appearance description, we develop an improved region covariance descriptor [8], which is able to adaptively extract visual features relevant to a specific target of interest. These visual features are fused in a covariance descriptor. We name the covariance descriptor, computed in this fashion, an "adaptive covariance descriptor" because the feature extraction method is adaptive to the specific objet to be tracked. In fact, the features are first projected to a principal subspace using principal component analysis, before computing the covariance descriptor. Then, in subsequent frames, pixel features in candidate regions are also projected in the target subspace, enhancing the distinguishability of the true target candidate. The rationale behind this idea is to use a combination of features which is automatically optimized to distinguish a specific target (colors, shape, correlation color/color, ...) by increasing the dissimilarity with potential region candidates. This procedure could be interpreted as a modification of the metric of the space of covariance descriptors according to the target to be tracked. The resulting adaptive covariance descriptor is not only more effective but also more efficient to compute.

Concerning the decision algorithm yielding the localization of the target, we propose to combine both merits of generative and discriminative models by organizing them in a cascaded detection structure. The generative and discriminative models are trained collaboratively in an online way in order to efficiently learn the target appearance. Furthermore, multiple models are integrated in order to enhance the tracking performance.

Finally, for updating the object appearance models, we develop a clustering-based method for updating the generative model(s) and an online PLS learning method for updating the discriminative model(s). An efficient implementation of the online PLS method is also proposed in this thesis.

We summarize the main contributions as following:

1. An ameliorated target-oriented region descriptor called adaptive covariance descriptor which is able to extract compact and relevant features fused in a covariance matrix descriptor for image region appearance representation.
2. An hybrid target detection algorithm by integrating multiple generative and discriminative models in a cascaded detection structure.
3. A weakly-supervised method using the mean-shift clustering for updating the generative object appearance model(s).
4. An online Partial Least Squares (PLS) model learning method which is able to incrementally or decrementally update a PLS-1[1] regression model within constant time and space complexities. A weighted version is developed as well, that can assign weights to the learned data and the newly acquired data when updating the model.

The global tracking system integrates all the proposed methods as building blocks, yielding a robust tracker that can handle most challenges involved in visual tracking. Our experimental results showed that the final tracking system can work quite well in difficult real-world video sequences and outperform several state-of-the-art methods.

1.4 Structure of the thesis

The structure of the thesis is organized as follows. In Chapter 2, we review the advances in appearance-based tracking approaches. In Chapter 3, we introduce the adaptive covariance descriptor and the clustering-based model updating method. In Chapter 4, we

[1] PLS-1 algorithm refers to the the PLS regression method where the response data is univariate.

introduce the online PLS model learning method. In Chapter 5, we build a robust tracking system that integrates generative and discriminative models in a cascaded structure. The tracker includes the adaptive covariance descriptor, the clustering-based updating method and the online PLS learning method as well. Finally, Chapter 6 gives a summary of thesis and the achieved results. It also introduces some possible future extensions and perspectives.

Chapter 2

Advances in Visual Tracking

The state of the art of visual tracking has significantly evolved in the past 30 years. In the early years, almost all visual tracking methods assumed that the object motion was smooth [9, 10] and that the target may not undergo large appearance variations [11]. Recently, tremendous progress has been made and some algorithms can deal with the problems of abrupt appearance variations [12], with situations where the target my leave and re-appear in the visual scenes [6] and the problems of drifting [13], etc.

Yilmaz et al. reviewed in [14] the object tracking methods before 2006, presenting detailed analysis and comparisons of various representative methods. Yang et al. reviewed recent advances and trends from 2006 to 2011 in [15]. More recently, a experimental survey is presented in [7] relating a few most influential trackers throughout history and some open-source trackers that appeared in major conferences and journals between 2011 and 2012.

As stated in Chapter 1, most visual tracking methods take the structure of four components: motion model, appearance description, similarity measurement within the detection algorithm and the model updating. We will present, in this chapter, more details concerning each component and review representative works in terms of the emphasized component(s). Some recent trends and most related works are also presented.

2.1 Motion model

When a new image frame arrives, one needs to make some assumptions of the possible location of the target. This is the motion model for a tracking system. Motion constraints restrict the space of states to be searched for the target. Motions models in existing tracking system generally fall into four categories, introduced in the following.

- The first category for motion is the implicit motion prediction as in Mean Shift Tracker [16], and KLT [17], which does not use any constraints on the motion model. Instead, it seeks the maximum by using optimization methods. However, it requires that the movements of the target are small relative to the appearance changes in the visual scene, which is rarely the case in general tracking.

- The most straightforward general motion model is an exhaustive sliding window approach where all possible locations in the frame are considered. This approach makes no assumption of the motion of the object, making it robust to wild object and camera motions. To handle object scale variation, a scale space is also exhaustively spanned. The problem of this motion model is the heavy computational burden, which makes it unpractical for real-time implementation. An improved approach is proposed in [18].

- A natural modification of the full-image search model is to search only in a local region around the previous position of the target. This local sliding-window approach also leaves the object entirely free in its motion pattern, hence is likely to be robust for many situations, but it may lose the target when its motion is quite fast. The model is adopted by a number of state-of-the-art tracking systems, e.g. [19][4][20], to name only a few.

- While the above models consider uniform search in the whole frame or in a local window, an alternative is the use of a probabilistic Gaussian motion model usually centered around the previous position as used in [21] and [22]. Moreover, due to the great success of Particle Filtering [23], also known as sequential Monte Carlo methods (SMC), visual tracking has been formulated in a Bayesian inference framework. Given a set of observed images $O_t = \{o_1, \cdots, o_t\}$, the inference aims at estimating the value of the hidden state variable Θ_t. Assuming a Markovian state transition and using the Bayes theorem, we have the following recursive equation

$$p(\Theta_t|O_t) \propto p(o_t|\Theta_t) \int p(\Theta_t|\Theta_{t-1}) p(\Theta_{t-1}|O_{t-1}) d\Theta_{t-1} \tag{2.1}$$

  where $p(\Theta_t|\Theta_{t-1})$ is the state transition model, $p(o_t|\Theta_t)$ is the observation model and $p(\Theta_{t-1}|O_{t-1})$ is the posterior probability recursively updated with time. The state estimate $\hat{\Theta}_t$ is then determined as the maximum a priori probability (MAP) estimate, i.e.

$$\hat{\Theta}_t = \Theta_t^{map} = \arg\max_{\Theta_t} p(\Theta_t|O_t). \tag{2.2}$$

Compared with the regular exhaustive search-based methods, the main advantage of the use of a particle filter is the reduction of sampling patches during tracking. Another benefit of the particle filter is that the sampling effort can be kept constant, independent

of the size of the object to track which is not the case with simply expanding the search region around the object with a fixed factor. Despite its great success, Particle Filtering often suffered from the curse of dimensionality [24] (very peaked likelihood), which is due to the suboptimal sampling techniques. Therefore, introducing more advanced Monte Carlo sampling methods would greatly elevate the visual tracking performance.

Some improvements of the basic Particle Filter motion model have been proposed. Zhou et al. [25] proposed adaptive velocity, adaptive noise and adaptive number of particles for a Particle Filter motion model. Kwon and Park proposed a Particle Filter on affine group using auto-regressive motion model, which can propose particles more effectively. Recently, prediction of motion as applied in Adaptive Coupled-Layer Tracker [27] may be helpful to counter full speed targets but it is not easily applicable in general tracking situations.

2.2 Appearance description

Appearance feature description plays a crucial role in visual tracking as the quality of the description directly relates to the quality of the tracking performance. In general, the most desirable property of a feature description is to make the object easily distinguishable against non-targets in the feature space.

From one pixel within a color image, the R, G, B color features can be naturally extracted. It is then not difficult to transform them into other color spaces or to gray levels. In addition, gradient and text features can also be extracted by considering the pixel within a local neighborhood.

In order to describe a region of pixels in a higher level, one popular way is to use a descriptor based on statistics, such as the histogram [28] and the covariance matrix [8] which have been widely used in many computer vision applications to represent the pixel feature distribution. The histogram descriptor is a nonparametric estimation of the distribution over pixels values in a region. It owns a simple form and shows good robustness against translation and rotation. It can be calculated efficiently, especially with accelerated algorithm like the integral histogram [29] or distributive histogram [30]. Although the histogram can accommodate any feature one at a time, the joint representation of several different features through histogram has an exponential load with the number of features. The covariance matrix [8], on the contrary, provides a natural way of fusing multiple features which might be correlated. It can integrate the spatial, color and gradient information all in one matrix and disclose the correlation among them. It is shown, in [8], that the covariance descriptor greatly outperforms

histogram descriptor for object detection and texture classification. Besides histogram and covariance matrix, there are essentially other representations, such as 2D-array like raw image data and feature vectors.

2.2.1 Feature descriptors

Gradient features characterize the shape information of the object and are often utilized in human detection. Gradient descriptors are the statistical summarizations of the gradients. For example, in [31], Lowe introduced the well-known SIFT descriptor for object recognition. Later, Bay et al. proposed SURF [32], which is a much faster scale and rotation invariant interest point descriptor. Dalal and Triggs [28] used the Histogram of Oriented Gradient (HOG) descriptor in training SVM classifier for pedestrian detection. Zhu et al. [33] improved its computational efficiency significantly by utilizing a boosted cascade of rejectors. Maji et al. [34] also demonstrated promising results using the multi-resolution HOG descriptor and the faster kernel SVM classification. Felzenszwalb et al. [35] described a part based deformable model based on the multi-resolution HOG descriptor for pedestrian detection.

Color descriptors have been proposed as well, showing robustness against certain photometric changes. The apparent color of an object is influenced primarily by two physical factors, (i) the spectral power distribution of the illuminance and (ii) the surface reflectance properties of the object. Recent advances in color descriptors can be categorized into novel histogram-based color descriptors and SIFT-based color descriptors. In the HSV color space, it is known that the hue becomes unstable near the gray axis. Weijer et al. [36] applied an error propagation analysis to the hue transformation. The analysis showed that the certainty of the hue is inversely proportional to the saturation. Therefore, the hue histogram is made more robust by weighing each sample of the hue by its saturation. The H color model is therefore scale-invariant and shift-invariant with respect to light intensity. The SIFT descriptor is not invariant to light color changes, because the intensity channel is a combination of the R, G and B channels. Weijer et al. [36] introduced a concatenation of the hue histogram with the SIFT descriptor, which is scale-invariant and shift-invariant. In [37], color invariants had been first used as an input to the SIFT descriptor, which leads to a CSIFT descriptor that is scale-invariant with respect to light intensity. More detailed performance evaluation of color descriptors can be found in [38] and [39].

Texture is a measure of the intensity variation of a surface, which quantifies properties such as smoothness and regularity. Gabor wavelet [40] is probably the most studied texture feature. The Gabor filters can be considered as orientation and scale tunable

edge and line detectors, and the statistics of these micro-features, in a given region, are often used to characterize the underlying texture information. In recent years, increasing interest is paid on investigating image local patterns for better detection and recognition. Especially, local patterns that are binarized with an adaptive threshold provide state-of-the-art results on various topics, such as face detection and image classification. In [41], Ojala et al. developed a very efficient texture descriptor, called Local Binary Patterns (LBP). The LBP texture analysis operator is defined as a grayscale invariant texture measure, derived from a general definition of texture in a local neighborhood. The most important property of the LBP operator is its tolerance against illumination changes. Another equally important characteristic is its computational simplicity. Mu et al. proposed in [42] two variants of LBP: Semantic-LBP and Fourier LBP. These new features can work in perceptually color space and prove more suitable for the human detection task. Inspired by Weber's Law, Chen et al. [43], developed a new local descriptor called the Weber Local Descriptor (WLD). It is based on the fact that human perception of a pattern depends not only on the change of a stimulus (such as sound, lighting) but also on the original intensity of the stimulus.

2.2.2 Multiple features fusion

Although tremendous progresses have been made, no single feature descriptor is robust and efficient enough to deal with all kinds of situations. For instance, the HOG descriptor focuses on edges and structures ignoring flat areas, but fails to deal with noisy edge regions. Color features represent the global information of images, which are relatively independent of the viewing angle, translation, and rotation of the objects and regions of interest. However, objects with the same color histogram may be completely different in texture, thus color histogram cannot efficiently characterize the object template. For the LBP descriptor, a possible drawback is that the thresholding operation when comparing the neighboring pixels could make it sensitive to noise.

To tackle this problem, feature combination has attracted more and more attention as it usually lead to boosted system performance and robustness. As previously mentioned, the covariance matrix descriptor [8] can encode the gradients strength, orientation and position information in covariance matrices. The main disadvantage of the covariance descriptor is that it is Symmetric Positive Definite (SPD) and thus lies on a Riemannian manifold. Operations through Riemannian geometry are usually time-consuming. Besides the design of new multiple features, some works show that using the combination of existing features can also improve the performance. In [44], Alahi et al. proposed a cascade of descriptors to detect and track objects through any network of cameras. Schwartz and Davis presented in [45] an efficient descriptor for pedestrian detection

based on Partial Least Squares (PLS) analysis. Such a descriptor includes the combination of gradient, texture and color information.

Recently, multiple kernel learning method has attracted increasing interest in computer vision community. Given multiple sources of information, one might calculate multiple basis kernels, one for each source. In such cases, the resulting kernel is often computed as a convex combination of the basis kernels. Kembhavi et al. [46] proposed an Incremental Multiple Kernel Learning (IMKL) approach for object recognition, which combines the Pyramidal Histogram of Oriented Gradients (PHOG) [47] and Geometric Blur [48] together. In [49], baseline feature combination methods, Multiple Kernel Learning (MLK) methods and ensemble methods inspired by Boosting are thoroughly evaluated on object classification datasets using a multitude of descriptors. It was found that even very simple baseline combination methods, which are much faster than MLK methods, achieved highly competitive performances. On the other hand, the Boosting type of methods produced consistently better results.

How to combine various kinds of features into a coherent framework still needs much more study. Besides, deeper understanding of human vision principles would also benefit feature descriptor research.

2.3 Object appearance model and similarity measure

The appearance representation implies a certain degree of constancy when transferring one frame to the next. Without any such constancy assumption, tracking cannot work. More precisely, it is assumed that the samples are generated from the same underlying probability distribution. Machine learning methods are then suitable and have been widely employed to fulfill visual tracking. The classifier learn to distinguish the target object based on its appearance model and a quantitative decision function.

When computing a classifier for object recognition, one faces two main philosophies, namely generative and discriminative models. Formally, the two categories can be described as follows. Given an input x and a label y, a generative classifier learns a model of the joint probability $p(x, y)$ and classifies using $p(y|x)$ which is obtained by using the Bayes rule. In contrast, a discriminative classifier models the posterior $p(y|x)$ directly from the data or learns a map from the input x to labels y: $y = f(x)$.

2.3.1 Generative models

For visual tracking, the background is "the rest of the world" except the target object, which is too wild to estimate its class conditional distribution. Therefore, most generative models in the literature only model the target object and totally ignore the background. In this sense, generative trackers represent the appearance of an object by learning a model that provides sufficient reconstruction ability. Tracking is expressed as finding the most similar object appearance to the model. As they model only the target object, techniques employed are generally unsupervised, such as Principal Component Analysis (PCA) [50], Independent Component Analysis (ICA) [51], Mixture Models [52], Expectation Maximization (EM) [53] and Compressive Sensing [54]. To handle the variability of a target, the object model is often updated online to adapt to appearance variations. In the following, we introduce some of generative methods for visual tracking.

- The first group of generative models uses mixture models for building object appearance models. Black et al. [55] employ a mixture model to represent and recover the appearance changes in consecutive frames. Jepson et al. [56] develop a more elaborated mixture model with an online EM algorithm to explicitly model appearance changes during tracking. Later, in [25], Zhou et al. embed the appearance adaptive models into a particle filter to achieve a robust visual tracking.

- Besides mixture models, online subspace learning is also widely employed in generative models. Li [57] propose an incremental Principal Component Analysis (PCA) algorithm for subspace learning. In [58], a weighted incremental PCA algorithm for subspace learning is presented. In [21], a generalized tracking framework based on the incremental image-as-vector subspace learning methods with a sample mean update is presented.

- Learning on manifolds is exploited as well. chih Lee and Kriegman [59] presented online learning of probabilistic appearance manifolds for video-based recognition and tracking. Porikli et al. [60] proposed a tracking framework using covariance matrix descriptor with mean update in Riemannian manifold. Based on the covariance matrix descriptor and the Log-Euclidean Riemannian metric [61], Li et al. [62] presented an online subspace learning algorithm which models the appearance changes by incrementally learning an eigenspace representation for each mode of the target through adaptively updating the sample mean and the eigenbasis.

- Furthermore, tensor learning is also employed. In [63], Li et al. present a visual tracking framework based on an online temporal tensor subspace learning. Later,

Wu et al. [64] presented a tracking approach that incrementally learns a low-dimensional covariance tensor representation, dealing with the temporal variations of target appearance.

- A recent trend of generative models base is inspired from recent advances in compressive sensing [54]. The l_1 tracker [65] obtains robustness by seeking a sparse representation of the tracked object via l_1 norm minimization. Since then, sparse representation for visual tracking has attracted an increasing interest. [66] extended the l_1-tracker by using the orthogonal matching pursuit algorithm for solving the optimization problems efficiently. More recently, [67] proposed an appearance model based on features extracted from the multi-scale image feature space with data-independent basis. A very sparse measurement matrix is adopted to efficiently extract the features for the appearance model. Both the target and the background are compressed using the same sparse measurement matrix. Finally, tracking is formulated as a binary classification via a naive Bayes classifier with online updating in the compressed domain.

The main problem of generative models is that they are prone to similar background regions (similar to the target) called "distracters", especially in cluttered scenes.

2.3.2 Discriminative models

For trackers that adopt discriminative models, a classifier is trained directly from training samples to find a decision boundary that best distinguishes the object from the background. This type of methods is aka "tracking-by-detection", where a target object, identified by the user in the first frame, is described by a set of features. A binary classifier separates the target from the background in successive frames.

Classification tools employed by discriminative methods are typically supervised techniques, such as Linear Discriminant Analysis (LDA) [68], Support Vector Machine (SVM) [69], Relevance Vector Machine (RVM) [70], Boosting [71], Random Forests [72], as well as their variants. When properly trained, discriminative methods can demonstrate robustness to avoid distracters in the background, in contrast to their generative counterparts.

Collins et al. [73] proposed a method to adaptively select color features that best discriminate the object from the current background. Similarly, Wang et al. [74] propose a tracking algorithm based on online selecting discriminative features from a large feature space with the Fisher discriminant method. Later, Grabner et al. [1, 75] designed an online boosting classifier that selects and maintains the best discriminative features from

a pool of feature candidates. Later, Saffari et al. [76] proposed an online Random Forest (RF) algorithm based on an online decision tree growing procedure. Compared with the online boosting method [1, 75], the RF method is more robust against label noise.

Avidan [77] extended the optical flow approach [78] with an SVM classifier for object tracking. Motivated by the SVM tracker, Williams et al. proposed a real-time tracker using sparse probabilistic regression via RVMs [70]. The RVM tracker builds a displacement expert, which directly estimates displacement from the target region. In addition, the system used an object detector in tandem, for object verification, possibly automatic initialization and recovery. In [19], the discriminative model maintains a set of discriminant functions each distinguishing one pattern in the object region from background patterns in the neighborhood. The discriminant functions are efficiently trained online using a differential version of Linear Discriminant Analysis (LDA). Object detection is performed by maximizing the sum of all discriminant functions. Later, Avidan [80] used an adaptive ensemble of classifiers for visual tracking. Each weak classifier is a linear hyperplane in an 11D feature space composed of R,G,B color and a histogram of gradient orientations. Tian et al. [81] presented an online ensemble linear SVM tracker, which makes good usage of history information during tracking.

In [82], Zhang et al. proposed a graph embedding based discriminative learning method, in which the topology structures of graphs are carefully designed to reflect the properties of the sample distributions. Wang et al. [83] proposed a beyond distance measurement for video annotation. In [84], multi-graph learning was used to unify video annotation.

Psychological and cognitive findings indicate that the human perception is attentional and selective. Inspired by this theory, Yang et al. [85] proposed a new visual tracking approach by reflecting some aspects of spatial selective attention, and presents a novel Attentional Visual Tracking (AVT) algorithm. The algorithm dynamically identifies a subset of discriminative attentional regions through a discriminative learning on the historical data on the fly.

2.3.3 Hybrid models

When applied separately for visual tracking, the discriminative methods are sensitive to label noise and generative methods are not effective for distinguish the target from its similar distracters. It has been shown that discriminative classifiers often outperform generative models [86] if enough training data are available. However, generative methods often have better generalization performance when the size of training data is small. For instance, [87] reported that a simple naive Bayes classifier (generative model) outperforms logistic regression (its discriminative counterpart) when the amount of labeled

training data is small. A number of hybrid approaches have been proposed aiming at fusing the advantages of both strategies. [88] describes a hybrid model where a high-dimensional subset of the parameters are trained to maximize generative likelihood, and another small subset of parameters are discriminatively trained to maximize conditional likelihood. In [89], Lin et al. train a model by optimizing a convex combination of the generative and the discriminative objective functions. [90] propose a principled combination of generative and discriminative models, showing that when the supply of labelled training data is limited, the optimum performance corresponds to a balance between the purely generative and the purely discriminative methods. In [91], Grabner et al. proposed a modified error function for boosting to select features that show good for both discrimination and reconstruction. In [92], Woodley et al. presented a tracking system using online discriminative feature selection guided by a local generative model. In [5], Yu et al. proposed to online co-train a global generative model and a local discriminative model for target tracking and reacquisition. The generative model uses a number of low dimension linear subspaces to describe the appearance of the object, which encodes all the appearance variations that have been seen in order to be able to reacquire the object. The discriminative classifier is implemented as an online support vector machine, which is trained to focus on recent appearance variations.

How to combine the generative machine learning methods and discriminative machine learning methods into a coherent framework is a classic question within machine learning field and also an interesting open question in the visual tracking literature.

2.4 Updating of the object appearance model

For visual tracking, handling appearance variations of a target object as well as its background is of great importance for tracking robustness. In general, there are two types of appearance variations: intrinsic and extrinsic. Pose variation and/or shape deformation of a target object are considered as the intrinsic appearance variations while the extrinsic variations are due to the changes resulting from different illumination, camera motion, camera viewpoint, and occlusion.

These variations can only be handled with adaptive methods which are able to incrementally update their object appearance models. To handle such variations, the object appearance model needs to be adjusted to the new circumstances from time to time. To handle appearance changes, the object appearance model is updated incrementally over time. Thus, there is an essential need for on-line algorithms that are able to learn continuously.

Despite its efficiency, online adaption faces one key problem: each update of the tracker may introduce an error which, finally, can lead to tracking failure. Most commonly, the foreground and background are divided by a bounding box or a region around the location of the object. No matter how tight the region is, such a partition is too rough because some background regions are treated as a part of the foreground, especially when the location of the object is not precise or the object is occluded. The adaptive tracking system will eventually degrade due to inaccuracy in the estimation of the foreground and background. This problem is called the Drifting Problem [13]. A closely related problem is how to achieve good balance between adaptivity and stability when using online learning methods.

To deal with these problems, a variety of efforts have been made. For instance, in [13], the "drifting" problem was firstly presented and a template updating methods with drifting correction is proposed. The drifting correction is based on the alignment of the preliminary results to the initial template, which makes it ineffective for situations where the appearance of the target changes significantly. Similarly, Grabner et al. [2] proposed a semi-supervised approach where labeled examples come from the first frame only, and subsequent training examples are left unlabeled. Although this method is well suited for scenarios where the object leaves the field of view completely, it is difficult to decide the exact object locations in the first frame. To obtain accurate boundaries of the tracked object and thus alleviate the drifting problem, Aeschliman et al. [93] proposed a novel probabilistic framework for jointly solving segmentation and tracking, which achieved significantly improvement in tracking robustness. An alternative is [4], which proposed a tracking-by-detection method based on the online Multiple Instance Learning (MIL) method. The MIL resolves the uncertainties of where to take positive updates during tracking, making the tracker robust to partial occlusion. Motivated by the merits of both semi-supervised methods [94] and multiple instance learning methods [4], Zeisl et al. [95] proposed an online semi-supervised learning algorithm which is able to combine both of these approaches into a coherent framework. This leads to more robust results than applying both approaches separately.

Recently, a trend to combine multiple trackers or to integrate trackers with detectors has shown to be promising for increasing the robustness of online updating. We present, in the following, some of theses promising directions.

- In [96], two classifiers with independent features are co-trained within online support vector machines. The predictions from different features are fused by combining the confidence map from each classifier using a classifier weighting method, resulting in a final classifier that performs better than any single classifier. Another example is Yu et al.. As the previously mentioned method, the method co-trains a

global generative model and a local discriminative model online. To enable reacquisition and recovery, the generative model uses a number of low dimension linear subspaces to encode all the appearance variations that have been seen.

- Some other methods to deal with the drifting problem use cascaded classifiers usually with distinctive thresholds. In [12], a cascade particle filter with discriminative observers of different life spans is applied. The observers employed different confidence thresholds in order to track in low frame rate videos. A disadvantage of this approach is that offline training is required to learn a long life-span observer. Along the same lines, Breitenstein et al. proposed in [97] a multi-person tracking-by-detection algorithm with a cascade detection confidence threshold mechanism, which aims at avoiding the errors introduced by the online learning classifier.

- In [98], Kwon and Lee proposed to sample multiple motion models and multiple appearance models and integrate them through an interactive Markov Chain Monte Carlo (IMCMC) framework. The overall observation model is decomposed into multiple basic observation models that are constructed by Sparse Principal Component Analysis (SPCA) of a set of feature templates covering a specific appearance of the object. The motion model is also represented by the combination of multiple basic motion models, each of which covers a different type of motion.

- Santner et al. [99] proposed a sophisticated tracking system called PROST that achieves top performance with a smart combination of complementary trackers. Three trackers of different degrees of adaptivity are combined: a simple template model based on normalized cross correlation [100] as a nonadaptive stable component; an optical-flow based mean-shift tracker [101] as highly adaptive element and an online random forest [76] as moderately adaptive appearance based learner.

- Later, Kalal et al. [6] proposed a Tracking-Learning-Detection (TLD) framework where a Median-Flow tracker (a pyramidal Lucas-Kanade tracker [102] extended with forward-backward error checking) is combined with online learned classifier. The highlight in the TLD framework is the learning component called P-N learning which exploits both temporal and spatial structure in a video to progressively improve the accuracy of the classifier.

- More recently, an ensemble framework is proposed in [103] for multi-target tracking that optimally chooses target tracking result from that of independent trackers and a detector at each time step. Optimal selection is achieved through a hierarchical data association step with parameters discriminatively trained from a max-margin framework.

Chapter 3

Improved Region Covariance Descriptors and Clustering-Based Model Updating

3.1 Introduction

In this chapter, we will develop a novel tracking system that consists of novel approaches for both region representation and object appearance model updating. For region appearance representation, we propose an improved region covariance descriptor, called adaptive covariance descriptor. For object appearance model updating, we propose a weakly-supervised method based on clustering analysis using mean-shift gradient density estimation. In the first part of this chapter, we introduce region appearance representation using improved covariance descriptors, which are ameliorated variants of the widely used region covariance descriptor [8]. A briefly review of the region covariance descriptor as well as its distance metrics are presented in Section 3.2. In Section 3.3, we propose some improvements to the conventional covariance descriptor. Effectiveness of these improvements is empirically evaluated in Section 3.3.5.

In the second part of this chapter, we propose a novel object appearance model updating method that exploits feature space analysis using mean-shift clustering. By combining the improved covariance descriptor and the clustering-based model updating method, we develop a preliminary tracking system, which is presented in Section 3.4 and evaluated in Section 3.4.4.

3.2 Review of the region covariance descriptor

We will make a brief review of the region covariance descriptor in Section 3.2.1 as well as its distance metrics and intrinsic means in Section 3.2.2. Properties and problems of the covariance descriptor are discussed in Section 3.2.3.

3.2.1 Region covariance descriptor

The region covariance descriptor was firstly proposed by Tuzel et al. in [8]. The idea is to represent a feature distribution using its sample covariance matrix.

Let I be a $W \times H$ one-dimensional intensity or three-dimensional color image, and F be the $W \times H \times d$ dimensional feature image extracted from I

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

where Ψ is a function extracting image features such as intensity, color, gradients, and filter responses, etc. For a given rectangular region $R \in I$, denote $\{f_i\}_{i=1,\ldots,N}$ as the d-dimensional feature points obtained by Ψ within R. The region R is then represented by a $d \times d$ covariance matrix:

$$C_R = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \mu)(f_i - \mu)^\top \tag{3.2}$$

where μ is the mean vector of $\{f_i\}_{i=1\ldots N}$.

For fast calculation of covariance matrices, [8] also provided an intermediate representation called integral image. With this representation, covariance descriptor of any rectangular region can be computed within constant time [8].

3.2.2 Distance metrics and intrinsic means of covariance matrices

Covariance matrices do not lie on the Euclidean space. Therefore, an arithmetic subtraction of two matrices would not measure the distance of the corresponding regions. In fact, nonsingular covariance matrices are Symmetric Positive Definite (SPD) and lie on a connected Riemannian manifold. Accordingly, Riemannian metrics should be used for computing distance and mean of covariance matrices.

There are two Riemannian metrics proposed in the literature. One is the affine-invariant Riemannian metric presented in [104] and [105]. The other is the bi-invariant Log-Euclidean metric introduced in [61]. Under the affine-invariant Riemannian metric,

distance between two covariance matrices is computed as

$$\rho(C_1, C_2) = \sqrt{\sum_{i=1}^{d} \ln^2 \lambda_i(C_1, C_2)} \tag{3.3}$$

where $\{\lambda_i(C_1, C_2)\}_{i=1\ldots d}$ are the generalized eigenvalues of C_1 and C_2 computed from

$$\lambda_i C_1 x_i - C_2 x_i = 0 \qquad i = 1 \ldots d \tag{3.4}$$

and $x_i \neq 0$ are the generalized eigenvectors. Under the affine-invariant metric, there is no closed-form solution for computing the *intrinsic mean* of multiple covariance matrices. By exploiting the Lie group structure of SPD matrices, [60] presented an iterative optimization procedure for computing the intrinsic sample mean.

Under the Log-Euclidean Riemannian metric, distance measure between covariance matrices preserves much of the natural properties of the affine-invariant metric while being computationally straightforward: the distance between two covariance matrices C_1 and C_2 is given by,

$$d(C_1, C_2) = \| \log(C_2) - \log(C_1) \|_{\ell_2}, \tag{3.5}$$

where $\| \cdot \|_{\ell_2}$ is the ℓ_2 vector norm (which is equivalent to the Frobenius norm in this case) and $\log(C)$ is the matrix logarithm of the square matrix C. In addition, the log-Euclidean mean of multiple covariance matrices $\{C_1, \ldots, C_n\}$ can be obtained in closed form as

$$\bar{C} = exp\left(\frac{1}{n}\sum_{i=1}^{n} \log(C_i)\right). \tag{3.6}$$

Clearly, distance and mean under the Log-Euclidean metric take a much simpler form than those under the affine-invariant metric. See more details and comparison of these two metrics in [61].

With these metrics, similarity measurement between two image regions can be simplified as distance of their corresponding covariance descriptors. Accumulated image patches can be represented by the mean of their covariance descriptors. Based on this, most computer vision applications, such as object detection, target tracking and texture analysis have been deployed [8, 60, 106].

3.2.3 Discussion

Most applications that employ the covariance descriptor compute the descriptor using a fixed set of features, which is often determined a priori. For instance, in [8], each pixel

is converted to a nine-dimensional feature vector for object detection:

$$f(x,y) = [x\ y\ R(x,y)\ G(x,y)\ B(x,y)\ |I_x(x,y)|\ |I_y(x,y)|\ |I_{xx}(x,y)|\ |I_{yy}(x,y)|]\,, \quad (3.7)$$

where R, G, B are the three color channels in the RGB color space, I denotes the pixel intensity and I_x, I_{xx}, I_y, I_{yy} are the first- and second-order image derivatives of I with respect to the Cartesian coordinates x and y respectively. This feature set remains unchanged in [8] for all kinds of objects, without considering the characteristics of each object.

Actually, color can be interpreted and modeled in different ways. With the availability of a variety of color spaces, e.g. RGB, HSV, YCrCb, YUV, CIE Lab, CIE Luv, etc., the inevitable question is how to select proper color models that can produce good performance for detecting a particular object. Likewise, the gradient features, which encode the shape information of the region context, can also have a variety of choices. In deed, they can be computed using different combinations of orders, and further with their corresponding magnitudes and orientations. Consequently, how to choose the feature set to be fused in the covariance descriptor for detection is of great importance.

A number of works [44, 107–109] have empirically studied the performances of the covariance descriptor using different feature sets. The reported results showed that significantly different performances were achieved when using different features. This further shows the importance of feature selection or extraction for the covariance descriptor. Alahi et al. [44, 107] compared different feature sets for detection and tracking objects across non-calibrated camera networks and claimed that increasing the number of features may increase the performance of covariance descriptor. In addition, Alahi et al. suggested that shape information is crucial for inter-category object detection. For instance, gradient features perform well in pedestrian detection applications because the shape of a human is a relevant cue to distinguish it from other objects, whereas color features perform best in intra-category classification cases such as object re-identification or recognition. In [108], Cortez-Cargill et al. constructed covariance descriptors with nine sets of features based on various color spaces. They obtained a best feature set which embeds many color channels from a variety of color spaces and reaches a performance of 99% for face detection. However, the feature vector they got turned out to be a 20-dimensional one and thus makes the construction and similarity measure of the covariance matrices rather time-consuming.

In brief, two points can be drawn. First, different feature combinations generally produce different detection performances. Second, previous works generally reported better results using more features. Subsequently, two questions naturally arise. First, how to select proper feature set for detecting a specific object to ensure good performance

in terms of detection accuracy? Second, is it always true that fusing more features produces better detection performance? If yes, are there alternatives that use compact feature set while ensuring good performance? If not, what is the condition when more features do not yield better performance? We will try to answer these questions in the next section by analyzing the generalization ability of the region covariance descriptor from a machine learning perspective.

3.3 Variants of the region covariance descriptor

3.3.1 A machine learning perspective of the region covariance descriptor for object detection

The essence of image region matching is to measure the similarity between the object template and a candidate image patch. Region descriptors using statistics of the pixel set are designed to represent feature distribution of the pixels inside an image region. As such, similarity between feature distributions are reduced to compare the distance of corresponding region descriptors. Object detection using region descriptors takes the underlying assumption that descriptors computed from image regions that contains the same object have smaller distances than those computed from non-targets. This is indeed a machine learning process. With a training set of the object template, one seeks to compute statistics to characterize the feature distribution of the object. Two typical statistics that represents feature distribution based on training samples are the histogram and the covariance matrix.

For object detection, one has a training set, i.e. the pixel set of the object template. Each pixel is represented by a vector of features that are extracted from the image. As such, statistical models can be learned from this training set in order to represent the object to be detected. In this sense, the region covariance descriptor estimates the covariance of the feature distribution using the training sample set. It then estimates variances of the features in the diagonal entries of the matrix and covariances between pairs of features in off-diagonal entries to represent the second order statistics of the pixel feature distribution. Testing a candidate sample is conducted by distance measure between corresponding region descriptors. For the region covariance descriptor, this can be done either using the affine-invariant metric or the Log-Euclidean metric. It is worth noting that both of the Riemannian metrics compute logarithm of eigenvalues in order to transform a point from the SPD Riemannian manifold into a local Euclidean space.

The detection performance of the descriptor depends mainly on the generalization ability of the model, which can be analyzed by means of the bias and variance decomposition.

First, the eigenvalues are estimated from limited training samples. It is well known that the estimates based on Equation (3.2) produces biased estimates of the eigenvalues; the largest ones are largely biased and the smallest ones are biased towards values that are too low. This bias is most pronounced when the population eigenvalues tend towards equality, and is correspondingly less severe when their values are highly disparate. In all cases, this phenomenon becomes more pronounced as the sample size decreases [110–112]. Second, if there are very small eigenvalues in the sample covariance matrix, logarithm of these tiny eigenvalues will incur large disturbance which can dramatically degrade the generalization ability.

Therefore, analysis of the eigenspectrum of the sample covariance matrix of the object template is of importance. If there are some very small eigenvalues, reduction of incurred variance is necessary. To this end, we propose in the subsequent two remedies for poorly-conditioned covariance matrices: one by regularization and the other by dimension reduction.

3.3.2 Regularized covariance descriptor

To cure the large variance problem caused by tiny eigenvalues, our first solution is to use regularization techniques, which have been highly successful in the solution of ill- and poorly-posed inverse problems. Specifically, we regularize the estimated covariance matrix by adding a scaled identity matrix to it, i.e.

$$C_R = C_R + \eta E \tag{3.8}$$

where E is the identity matrix that has the same size as C_R. With sufficiently large η, this regularization can effectively cure the poorly-conditioned sample covariance matrix. We name the resulting region covariance descriptor after regularisation the "regularized covariance descriptor".

Regularization reduces the variance associated with the sample based estimate at the expense of potentially increased bias. Hence, the choice of the value of η is of importance. Generally, over-regularization using large η will introduce large bias whereas under-regularization will not effectively cure the large variance problem. To determine a proper value for η, one needs to take into account several factors, e.g. the slope of the log function, the range of the feature channels, among others.

3.3.3 Adaptive covariance descriptor

Another way to reduce variance caused by tiny eigenvalues is to use PCA projection to remove those unreliable dimensions while preserving dominant information in the principal components.

Specifically, we first extract raw features from the image patch to form a set of n d-vectors (n indicates the number of pixels in this region and d is the dimension of features); based on this point set, we not only construct the original covariance matrix, but also learn a PCA projection. The d-dimensional data set is then projected to a subspace by the learned PCA projection yielding a compact k-dimensional point set. Finally, the adaptive covariance matrix descriptor is constructed using the projected point set.

For a candidate image patch to be compared with the template, the descriptor computation is similar except that it employs the PCA projection pre-learned from the template point set. In this way, the feature extraction is adaptive to a specific target. We name the region covariance descriptor computed in this fashion the "adaptive covariance descriptor".

3.3.3.1 Computation of the adaptive covariance descriptor

In the training stage, based on a d-dimensional feature pool and the point set from the object template image, the PCA projection matrix is learned by keeping the k ($1 \leq k \leq d$) top eigenvectors of the sample covariance matrix according the significance of their corresponding eigenvalues. The mean vector of the training samples is preserved as well.

When generating the adaptive covariance descriptor, each point represented by $f(x, y)$ is firstly subtracted by the mean of the training samples. Then, it is projected to the subspace spanned by the k retained eigenvectors, yielding a compact k-dimensional feature vector $p(x, y)$. Finally, the adaptive covariance descriptor of an image region is computed using the sample covariance of the extracted feature vector $p(x, y)$

$$C_{a,R} = \frac{1}{N-1} \sum_{i}^{N} p_i \times p_i^{\top} \tag{3.9}$$

We summarise the procedure in Algorithm 1. Note that since the adaptive covariance descriptor is still a covariance matrix, the integral image [8] can be naturally inherited for fast covariance matrices computation.

Algorithm 1 Procedure for computing the adaptive covariance descriptor.

Training Stage:

Input: target template image from the initial frame

Output: $\mu(f)$: the mean feature vector;
V_k: the projection matrix.

1: Form the pixels that are inside the template image.
2: Extract feature vector $f_i \in \Re^{d\times 1}$ for each pixel i.
3: Compute the mean vector: $\mu(f) \leftarrow mean(f_i)$.
4: Compute the covariance matrix: $C_R \leftarrow cov(f_i)$.
5: Do eigenvalue decomposition for C_R: $C_R = V\Lambda V^\top$.
6: Keep the $k\,(0 \le k \le d)$ eigenvectors $v_{i=1\cdots k}$ in V that correspond to the k most significant eigenvalues:
$V_k \leftarrow [v_1 \ \cdots \ v_k]$

Generating Descriptors:

Input: any image region R, $\mu(f)$, V_k

Output: $C_{a,R}$: the adaptive covariance descriptor of the region R

1: Form the pixels that are inside the region R.
2: Extract feature vectors f_i for each pixel i.
3: Perform the PCA projection on each f_i and obtain a compact score vector $p_i \in \Re^{k\times 1}$: $p_i \leftarrow V_k^\top(f_i - \mu(f))$.
4: Compute the adaptive covariance descriptor $C_{a,R} \in \Re^{k\times k}$ using the sample covariance matrix of p_i: $C_{a,R} \leftarrow cov(p_i)$.

It is pointed out in [8] that for the conventional covariance descriptor, given a region R, its covariance C_R does not have any information regarding the ordering and the number of points, which implies a certain scale and rotation invariance over the regions in different images. However, if the matrix fuses the information regarding the orientation of the points, such as the norm of gradient with respect to x and y, the covariance descriptor is no longer rotationally invariant. The same argument is also correct for scale and illumination. The features fused in the adaptive covariance descriptor are linear combinations of the raw features. Therefore, its invariance property is the same as the conventional descriptor. That is, if the raw features are scale, rotation or illumination invariant, then the adaptive covariance descriptor is also scale, rotation and illumination invariant. Otherwise, invariance does not hold.

3.3.3.2 Relation to the conventional covariance descriptor

We explore in this section the relationship between the conventional covariance descriptor and the proposed adaptive covariance descriptor in order to elucidate the superior representation ability of the proposed descriptor.

Let C_r denote the conventional covariance matrix descriptor computed for the target template image (the reference) and C_c denote that of an arbitrary candidate image region to be matched with C_r. Distance between C_r and C_c under the Log-Euclidean metric [61] is written:

$$d(C_r, C_c) = \| \log(C_r) - \log(C_c) \|. \tag{3.10}$$

After the PCA projection, the adaptive covariance descriptor for the target template becomes

$$C_{a,r} = V_k^\top C_r V_k. \tag{3.11}$$

Similarly, the adaptive covariance descriptor for the candidate image becomes

$$C_{a,c} = V_k^\top C_c V_k. \tag{3.12}$$

The distance between the $C_{a,r}$ and $C_{a,c}$ is thus written:

$$\begin{aligned} d(C_{a,r}, C_{a,c}) &= \| \log(V_k^\top C_r V_k) - \log(V_k^\top C_c V_k) \| \\ &= \| V_k^\top \cdot (\log(C_r) - \log(C_c)) \cdot V_k \|. \end{aligned} \tag{3.13}$$

It is interesting to find that if all the eigenvectors are kept, i.e. $k = d$, the new distance in (3.13) is equal to the original distance (3.10). This equality indicates that rotation of coordinate systems using PCA projection does not affect distances between covariance matrices. Nevertheless, this rotation by PCA projection is of interest, because it provides a most "suitable" coordinate system for analyzing the fused features from the perspective of the target template image.

In other cases where only a few principal components are kept, the equality of (3.13) and (3.10) no longer holds. The dimension reduction of PCA removes unreliable dimensions and thus makes the adaptive covariance descriptor different from the conventional one. From a machine learning perspective, the PCA projection during the computation of the adaptive covariance descriptor preserves dominant information and removes noise. Removing the unreliable dimensions can alleviate the overfitting problem and hence improve generalization.

Compared to the conventional descriptor, another advantage of the adaptive descriptor is that it is more compact. As such, it can make the subsequent operations, e.g. distance measure and appearance model updating, much faster. We acknowledge that the adaptive covariance descriptor may impose additional computational burden for training the PCA and projecting the raw feature vectors into principal components. However, this additional computational effort for obtaining a compact representation is well worth it. Firstly, training is an offline process, which is performed only once at the first frame.

The computation during the training stage for learning the PCA is thus negligible. Secondly, the benefit of the compact representation in subsequent processing may outweigh the cost of the PCA projection. In the practice of object detection, hundreds of thousands of candidate descriptors need to be computed and be compared for a test image. As such, employing the compact representation of the adaptive descriptor can result in significant efficiency gain.

In brief, the adaptive covariance descriptor represents an object using a covariance matrix that fuses a few relevant features formed by the principal components of the raw feature distribution. Compared to the conventional descriptor, operations on the adaptive descriptor are generally faster. Furthermore, if the conventional descriptor has very small eigenvalues, the adaptive descriptor should have better generalization ability than the conventional one.

3.3.4 ℓ_1 norm for distance measure

Since the logarithm domain of the SPD matrices manifold is in Euclidean space [61], one may consider using ℓ_1 norm instead of the ℓ_2 norm to measure the distance of two matrices in the logarithm domain. The intuition is that the ℓ_1 is generally more robust to outlier than the ℓ_2 norm [113]. We can thus expect the modified distance metrics using ℓ_1 norm to yield better detection performances.

Specifically, the two Riemannian metrics are modified as follows. For the Affine-Invariant metric, distance between two covariance matrices are modified as

$$\rho(C_1, C_2) = \sum_{i=1}^{d} |(\lambda_i(C_1, C_2)| \tag{3.14}$$

where $|\cdot|$ is the abstract value function. Likewise, the Log-Euclidean metric can be modified using ℓ_1 norm as

$$d(C_1, C_2) = \| \log(C_1) - \log(C_2) \|_{\ell_1} \tag{3.15}$$

where $\|\cdot\|_{\ell_1}$ is the ℓ_1 norm by taking the matrix $\log(C)$ as vector.

3.3.5 Empirical Evaluation

In order to validate the effectiveness of the claimed improvements to the conventional region covariance descriptor, we empirically assessed the performances of the regularized descriptor and the adaptive descriptor in comparison with that of the conventional

descriptor by repeatedly detecting objects in real-world video sequences.

As a benchmark, 3 publicly available sequences, namely "PkTest01", "PkTest02" and "PkTest03", from the VIVID airborne sensor dataset[1] [114] were used for evaluation. The three sequences from VIVID dataset are thermal infrared data of vehicles captured by moving cameras in airport circumstances. These sequences are selected because there is strong correlation between the color channels. Therefore, if many color features are used, there will be some very small eigenvalues in the covariance matrix. In addition, there are similar vehicles in the scene, making the detection challenging. As the sequences are very long, we used only the first 100 frames of each sequence. In addition, a public color sequence from the PETS dataset[2] is used as well. The sequence is captured from a static camera in a campus circumstance, where we seek to detect a walking pedestrian. There are some other pedestrians in the scene. Similar to the VIVID sequences, we used only 100 frame (from Frame #1412 to Frame #1511) of the sequence. Figure 3.1 displays the target objects, marked in rectangles in the first frames.

(a) PkTest01 (b) PkTest02 (c) PkTest03 (d) PETS

FIGURE 3.1: Initial frame of each sequence with target marked in rectangle.

3.3.5.1 Settings

Using the annotated template image in the first frame, we first computed the three descriptors, i.e. the conventional descriptor, the regularized descriptor and the adaptive descriptor of the object respectively. As in [8], an object was represented by five covariance matrices of the image features computed from five subregions (the whole region, the left half part, the right half part, the top half part and the bottom half part) of the object template image. Then, we used each descriptors to detect the object in the rest of the sequence and evaluated its performance.

[1] Available at http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html.
[2] Available at http://ftp.pets.rdg.ac.uk/PETS2001/DATASET1/TRAINING/CAMERA1_JPEGS/.

For all the sequences, we used a feature set $f(x, y)$ defined as

$$f(x,y) = \Big[R(x,y)\ \ G(x,y)\ \ B(x,y)\ \ H(x,y)\ \ L(x,y)\ \ S(x,y)\ \ a(x,y)\ \ b(x,y)\ \ u(x,y)\ \ v(x,y) \\ \frac{\partial I(x,y)}{\partial x}\ \ \frac{\partial I(x,y)}{\partial y}\ \ \frac{\partial^2 I(x,y)}{\partial x^2}\ \ \frac{\partial^2 I(x,y)}{\partial y^2}\ \ \frac{\partial^2 I(x,y)}{\partial x \partial y}\ \ \frac{\partial^3 I(x,y)}{\partial x^2 \partial y}\ \ \frac{\partial^3 I(x,y)}{\partial x \partial y^2}\ \ \frac{\partial^4 I(x,y)}{\partial x^2 \partial y^2}\Big]^\top \tag{3.16}$$

where $H(x, y)$, $L(x, y)$ and $S(x, y)$ are the feature channels from the HLS color space. Similarly, $a(x, y)$, $b(x, y)$ and $u((x, y)$, $v(x, y)$ are from the CIE Lab and CIE Luv color spaces respectively. The L channels in Lab and Luv colors spaces are not used because they are highly correlated with each other and also with the L channel in the HLS space. Note that all the color channels need to be adjusted to fall into the range of $[0 - 255]$. The derivatives of the intensity image are computed as they are using the Sobel operator with 3×3 or 5×5 kernels[3].

The conventional descriptor used this set directly. The regularized descriptor used the same feature set $f(x, y)$. The parameter η was set to 0.5. For computing the adaptive descriptor, the number of retained principal components after PCA projection was automatically determined. Those dimensions with corresponding eigenvalues less than 0.01 were removed.

The search method for locating the object is also similar to that in [8]. Initially, we computed only the descriptor of the whole region. We search the target image for a region having similar covariance matrix. Search was performed by sliding-window from left to right and from top to bottom in the whole image frame. The window size was fixed as that of the template image with no scale change. The search window jumped horizontally 10% of the width or vertically 10% of the height of the object between two search locations. After this first phase, we kept the best matching 1000 locations. At the second phase we repeated the search in 1000 detected locations, using all the five covariance matrices. The dissimilarity of the object and a candidate region was computed by summarizing the distances of all five pairs of covariance matrices. Finally, the region with the smallest distance was selected as the matching region. We used the Log-Euclidean metric to measure the distances between covariance matrices. Our implementation is in C++ and is based on the OpenCV library.

Two quantities were measured to evaluate the performance of the descriptors. One is the detection rate, which is defined as the ratio of the number of frames where object location is accurately estimated to the total number of frames for detection. The detection result is considered to be accurate if the center position of the best match is within the 9×9 pixel neighborhood of that of the ground truth. The other metric is the average processing time per frame, employed to evaluate the computational efficiency.

3.3.5.2 Results

Table 3.1 summarizes the detection rates and the average processing time per frame using the conventional covariance descriptor (denoted as "Cov"), the adaptive covariance descriptor (denoted as "AdpCov"), and the regularized covariance descriptor (denoted as "RegCov").

[3] According to the summarised order of the partial derivatives, i.e. if the summarised order is less than 3, we used the 3×3 kernel; otherwise the 5×5 kernel was used.

For comparing the ℓ_1 norm and the ℓ_2 norm, performances using each norm are presented respectively. As such, in Table 3.1, "Cov1" denotes the utilization of conventional covariance descriptor using the Log-Euclidean metric with ℓ_1 norm, i.e. Equation (3.15), whereas "Cov2" denotes the utilization of the Log-Euclidean metric with the ℓ_2 norm. Other notations are similar. Note that the average processing time is the total average of both ℓ_1 and ℓ_2 norms.

TABLE 3.1: Detection performance comparison using the conventional covariance descriptor, the regularized covariance descriptor and the adaptive covariance descriptor. Frames where the object is severely occluded are not counted in the performance computation.

Sequence	Detection Rate						Time Per Frame in Seconds		
	Cov1	Cov2	RegCov1	RegCov2	AdpCov1	AdpCov2	Cov	RegCov	AdpCov
PkTest01	84%	83%	100%	91%	100%	91%	1.614665	1.50845	0.42641
PkTest02	69%	69%	91%	75%	82%	68%	1.798325	1.173205	0.48842
PkTest03	100%	78%	100%	88%	100%	90%	2.199425	2.005465	1.043075
PETS	100%	98%	100%	100%	100%	98%	4.02714	2.88185	2.952495

On the VIVID sequences, there were a number of very small eigenvalues in the eigenspectrum of the conventional descriptor. The detection rates clearly showed that these small eigenvalues degraded the performance of the conventional descriptor. The regularized descriptor and the adaptive descriptor both handled this problem effectively and boosted the detection rates. Besides, the ℓ_1 norm generally outperformed the ℓ_2 norm. In terms of efficiency, benefited from the reduced dimensionality, the adaptive descriptor was significantly faster than the other two descriptors. Therefore, if efficiency is a major concern, the adaptive covariance is indeed a good choice. We display some detection results detected by each descriptor on the "PkTest01" sequence in Figure 3.2 and those on the "PkTest02" sequence in Figure 3.3 respectively. Since the ℓ_1 norm generally performed better than the ℓ_2 norm, the presented results are those detected by the ℓ_1 norm.

On the PETS sequence, there were no very small eigenvalues in the eigenspectrum of the conventional covariance descriptor of the target object. Indeed, all the eigenvalue were greater than 0.1. Therefore, no dimensions were removed for the adaptive covariance descriptor. In terms of accuracy, we see that all the descriptors performed quite well with detection rates from 98% to 100%. The slight performance deterioration is due to a partial occlusion in the 84^{th} and 85^{th} frames (Frame #1496 and Frame #1497 in the original sequence). The ℓ_1 norm successfully overcame this problem for all the three descriptors while the ℓ_2 norm drifted to another pedestrian in the scene except for the regularized descriptor. This phenomenon is displayed in Figure 3.4, which further evidenced that the ℓ_1 norm is more robust to outliers than the ℓ_2 norm. In terms of efficiency, as dimensionality was not reduced, the adaptive descriptor was slightly slower than the regularized descriptor due to the extra computational burden of the PCA projection. A mysterious observation is that on this sequence the conventional descriptor was much slower than the other two descriptors. Similar phenomenon can also be noticed on the three VIVID sequences: the regularized descriptor was generally faster than the conventional descriptor. A plausible reason is that when the covariance matrices are poorly-conditioned (even though the descriptor of the target object is not poorly-conditioned, there may be poorly-conditioned descriptors among the numerous candidate regions), the implementation routine that performs

(a) Some detection results using the conventional covariance descriptor on the "Pktest01" sequence.

(b) Some detection results using the regularised covariance descriptor on the "Pktest01" sequence.

(c) Some detection results using the adaptive covariance descriptor on the "Pktest01" sequence.

FIGURE 3.2: Some detection results on the "PkTest01" sequence (a) using the conventional covariance descriptor, (b) using the regularized covariance descriptor and (c) using the adaptive covariance descriptor. The initial frame with the target object marked in rectangle is shown in Figure 3.1(a).

the matrix logarithm operation conducts some extra computation to improve stability and thus makes the conventional descriptor less efficient.

3.3.5.3 Discussion

The above experiments shows that small eigenvalues indeed degrade the generalization ability of the conventional covariance descriptor. In general, a fixed feature set cannot always work well in all circumstances. The analysis of the eigenspectrum of the conventional covariance descriptor is thus important for detecting the small eigenvalue problem. Once detected, the proposed two variants are both effective to cure this problem.

Another perspective of the adaptive covariance descriptor is that it uses PCA to extract relevant compact features that are adaptive to a specific object. Irrelevant features are discarded. On the contrary, the regularised descriptor tries to alleviate the adverse effect of the irrelevant features, with the relevant features almost unaffected.

It is also interesting to draw an analogy between the two variants of the covariance descriptor and those of the linear least squares regression. For regression from X to Y, the ordinary least squares (OLS) solution is $(X^\top X)^{-1}X^\top Y$. When $X^\top X$ is ill- or poorly-conditioned, one can

(a) Some detection results using the conventional covariance descriptor on the "Pktest02" sequence.

(b) Some detection results using the regularised covariance descriptor on the "Pktest02" sequence.

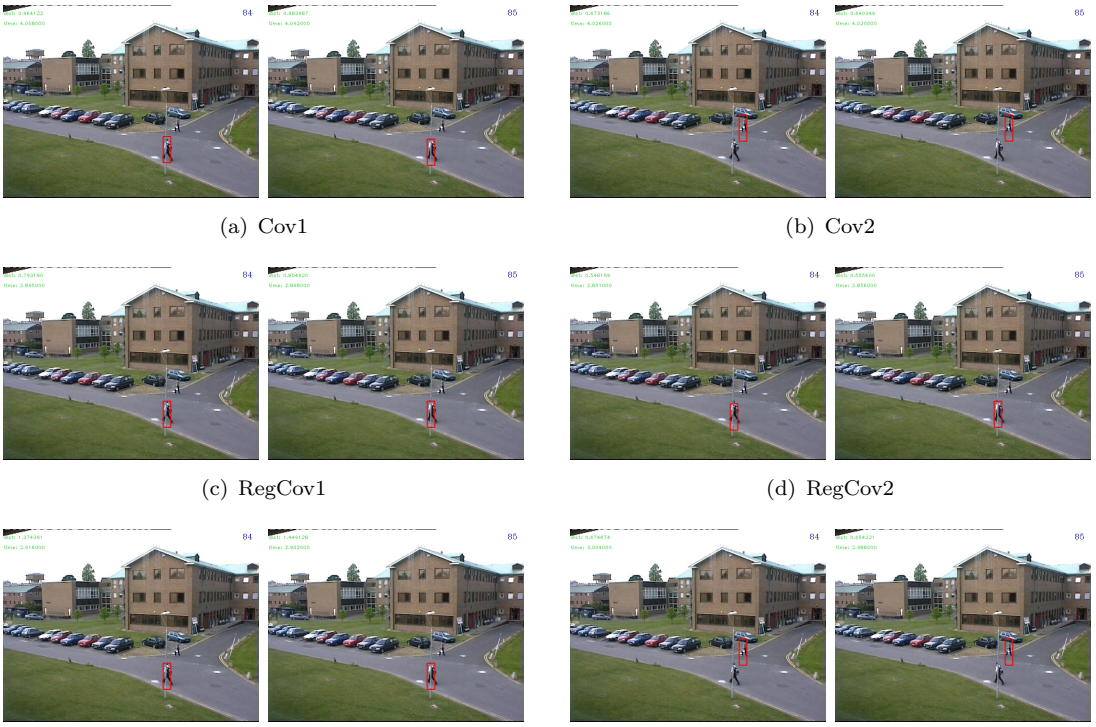
(c) Some detection results using the adaptive covariance descriptor on the "Pktest02" sequence.

FIGURE 3.3: Some detection results on the "PkTest02" sequence (a) using the conventional covariance descriptor, (b) using the regularized covariance descriptor and (c) using the adaptive covariance descriptor. The initial frame with the target marked in rectangle is shown in Figure 3.1(b).

either use the ridge regression or use the principal component regression (PCR) to handle this problem. The rigid regression is analogous to the regularized descriptor here and the PCR is analogous to the adaptive descriptor. This correspondence can be established because both the function $y = 1/x$ and the function $y = \log(x)$ have sharp slopes when x is very close to zero, which make the results unstable.

3.4 Object tracking

In this section, we shall build a tracking system that integrates the newly proposed adaptive covariance descriptor and a new model updating method. First, the object appearance model using multiple patches is presented in §3.4.1. Second, target localization using the appearance model and similarity measure is addressed in §3.4.2. Most importantly, for updating the model during the tracking, we propose in §3.4.3 a weakly-supervised updating method which is based on clustering analysis using the mean-shift procedure.



(a) Cov1 (b) Cov2

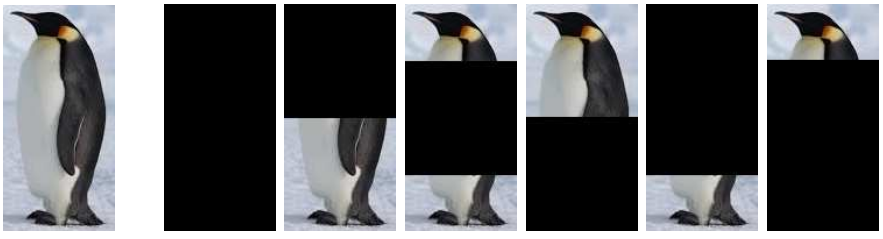
(c) RegCov1 (d) RegCov2

(e) AdpCov1 (f) AdpCov2

FIGURE 3.4: Detection results in the 84^{th} and the 85^{th} frames of the "PETS" sequence. See text for details. The initial frame with the target marked in rectangle is shown in Figure 3.1(d).

3.4.1 Object appearance model

To increase robustness, we use multiple patches of an image region, each of which is described by an adaptive covariance descriptor. A simple heuristic is employed to divide the object into six parts. If the width of the object is smaller than the height, the object is divided in the vertical direction. Otherwise, it is divided in the horizontal direction. This multi-part representation mechanism is illustrated in Figure 3.5, where an object on the left is vertically divided into parts on the right because its height is greater than its width. As such, a region is represented



(a) (b) (c) (d) (e) (f) (g)

FIGURE 3.5: Illustration of the multiple-patches object representation. An object on the left is represented by 6 adaptive covariance descriptors computed from corresponding subregions on the right. Note that if the width of the object is greater than its height, a similar division is performed in the horizontal direction.

by 6 adaptive covariance matrices computed from its 6 subregion patches. Each patch is then represented by an adaptive covariance descriptor, denoted as $\{C_a^i\}_{1\leq i\leq 6}$. For instance, when the height is greater than the width, C_a^1 is computed from the entire region as in Figure 3.5(b); C_a^2 from the top half as in Figure 3.5(c); C_a^3 from the middle half as in Figure 3.5(d); C_a^4 from the bottom half as in Figure 3.5(e); C_a^5 from the top 3/4 part as in Figure 3.5(f) and C_a^6 from the bottom 3/4 part as in Figure 3.5(g). For objects that have width greater than height, correspondence can be naturally established.

Although computed in a subspace, the adaptive covariance descriptor is indeed a sample covariance matrix, which lie on a Riemannian manifold. Using the Log-Euclidean transformation [26, 61, 62], we first transform the adaptive descriptors of the 6 patches $C_a^i(i = 1\cdots 6)$ to Euclidean space as $\log C_a^i(i = 1\cdots 6)$, then unfold each matrix and concatenate them to take a vector form. Note that since the transformed matrices $\log C_a$ are still symmetric, only upper triangular matrices are used. For instance, if 10 out of 15 dimensions are retained when computing the adaptive covariance descriptor, the dimension of the final vector representation of a region is $10 \times (10+1)/2 \times 6 = 330$, whereas the conventional covariance descriptor using 15 features would generate a vector of size $15 \times (15+1)/2 \times 6 = 720$.

Using the adaptive covariance descriptor and the transformations above, we obtain a vector-form feature representation of the target object, denoted as M_r, as the object appearance model. The appearance model learned from the initial target template image, denoted as M_r^0, is preserved throughout the tracking process for later participating the updating of M_r.

3.4.2 Target localization

To track the target in consecutive frames, we use an uniform sliding-window search around the target's previous position. That is, our motion model is such that the location of the tracker at time t is equally likely to appear within a rectangle window around the tracker location at time $(t-1)$. Let $\ell^*(t-1)$ denote the tracker location at time $(t-1)$, $x(\ell_{t-1}^*)$ be the x coordinate of $\ell^*(t-1)$ and $y(\ell_{t-1}^*)$ be the y coordinate of $\ell^*(t-1)$. The motion model is formally defined as

$$p(\ell_t|\ell_{t-1}^*) \propto \begin{cases} 1 & \text{if} \|x(\ell_t) - x(\ell_{t-1}^*)\| < s_1 \text{ and } \|y(\ell_t) - y(\ell_{t-1}^*)\| < s_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

where s_1 and s_2 are predefined constants that constrain the boundaries of the searching area.

At time t, when a new image frame is captured, a number of candidate regions are generated according to the motion model (3.17). Similar to the target object, each candidate region is represented using 6 adaptive covariance descriptors and then transformed to a vector-form feature representation. We denote the feature representation of the i-th candidate region as $M_{c,t}^i$. Distance between $M_{c,t}^i$ and the current target appearance model M_r is measured by

$$d(M_r, M_{c,t}^i) = \|M_r - M_{c,t}^i\|_{\ell_1}, \tag{3.18}$$

where $\|\cdot\|_{\ell_1}$ is the ℓ_1 vector norm.

The best match is the candidate region whose feature representation $M^*_{c,t}$ has the smallest distance to M_r, i.e.

$$M^*_{c,t} = \arg\min_i d(M_r, M^i_{c,t}). \tag{3.19}$$

The position of this best matching region then determines the location of the object, $\ell^*(t)$, in the current frame. Besides, $M^*_{c,t}$ is retained in a buffer for later updating the object appearance model M_r.

3.4.3 Weakly-supervised model updating

As time progresses, the target object may undergo both intrinsic and extrinsic variations. Updating of the appearance model M_r is thus necessary. An important issue for model updating is to ensure that the model is updated with correctly labeled samples. Contaminating the model with background samples will result in the well-known "drift" problem. Actually, tracking results collected during a certain period may contain optimal positive samples but also can have suboptimal or background samples. Previous work usually neglects this issue or simply address it by selecting good samples using a pre-fixed threshold, e.g. [25]. That is, updating is performed with samples which have distances to the object model smaller than a predefined threshold. However, during a long-term visual tracking, appearances of both the background and the target object are ever-changing. It is very difficult (if not impossible) to estimate a threshold that can separate optimal sample and suboptimal samples effectively in a long time.

To tackle this problem, our model updating method is based on two key observations that are obtained in the practice of visual tracking. First, an appearance model can effectively represent the target appearance for a certain duration. This indicates that with relatively robust appearance representation, it is not necessary to update the object appearance model too frequently. Second, in some cases there are no appropriate positive samples for updating the model. This usually happens in an occluded/absent scene where there is no "good" image region that contains the target object in that frame.

Based on the above observations, we propose to update the object appearance model by following a relatively long cycle, e.g. every 10 frames, instead of updating at each frame. Our idea is that a clustering analysis among the collected samples can naturally align similar optimal samples, suboptimal samples and background samples into different groups. The clustered group whose centroid is most close to the current object appearance model is selected for updating the object appearance model. As such, we can not only keep the appearance model adaptive to the changes but also prevent it from contamination when the tracker makes accidental tracking mistakes.

3.4.3.1 Mean-shift clustering for sample selection

As stated in §3.4.2, the tracking result sample estimated at each frame, i.e. $M^*_{c,t}$, is saved in a buffer which constitutes a sample set during a period of time. When the pre-fixed cycle is due, a clustering analysis is performed in the feature space among these samples. In practice, the concatenated vector representation is high dimensional. A PCA dimension-reduction procedure

can be performed in advance to obtain compact representation while preserving dominant information. In fact, Ding and He proved in [115] that principal components are the continuous solutions to the discrete cluster memberships indicators for K-means clustering. It is plausible that clustering in the projected subspace may improve the clustering accuracy [115, 116].

Since the tracker may occasionally make mistakes, the collected sample set can be any combination of optimal samples, suboptimal samples and background samples. It is thus very difficult to predict the number of clusters that are present. Hence, a standard clustering approach such as K-means is not appropriate. The mean shift clustering algorithm [117], which is an iterative gradient ascent method for finding local density maxima, was used instead. It does not require prior knowledge of the number of clusters and does not constrain the shape of the clusters. The data association criteria is based on the underlying probability of the data points.

The algorithm begins by placing a window (actually a hyper-sphere) around each point in the feature space. On each iteration, each window moves in the direction of the mean shift vector which is computed as follows:

$$y_{t+1} = \frac{1}{|\Theta_\lambda|} \sum_{x \in \Theta_\lambda} (y_t - x) \tag{3.20}$$

where y_t is the window center at iteration t, and Θ_λ is the set of points in the hyper-sphere window of radius λ. It is also possible to use a kernel function to weight points according to how far they are from the window center. The windows eventually converge towards local density maxima yielding the cluster centroid. The points that converges to the same local maxima naturally fall into the same cluster. As such, the mean shift clustering algorithm avoids the issue of knowing the number of clusters at the price of introducing another bandwidth parameter λ. This parameter, however, is intuitive and easy to tune regarding all possible inputs [118]. An example is presented in Figure 3.6 to illustrate the sample clustering process.

After clustering, the arithmetic mean of each cluster is computed. Among these means, the one $\bar{M}_s$ that has the smallest distance to M_r according to Equation (3.18) is selected for later updating M_r.

3.4.3.2 Updating of the object appearance model

As stated in §3.4.1, M_r^0 is preserved throughout the tracking. The updated object appearance model $\hat{M}_r$ is determined as a linear combination of M_r^0, M_r, and $\bar{M}_s$, i.e.

$$\begin{aligned} &\hat{M}_r = \alpha \cdot M_r^0 + \beta \cdot M_r + \gamma \cdot \bar{M}_s, \\ &s.t. \qquad \alpha + \beta + \gamma = 1.0; \qquad 0 \leq \alpha, \beta, \gamma \leq 1.0. \end{aligned} \tag{3.21}$$

Finally, the model updating is accomplished by setting

$$M_r = \hat{M}_r. \tag{3.22}$$

The advantage of employing the mixture coefficients, i.e. the α, β and γ, is that they can increase the flexibility of the model. Specially, with α set to 1.0, the model is kept fixed at M_r^0 and no

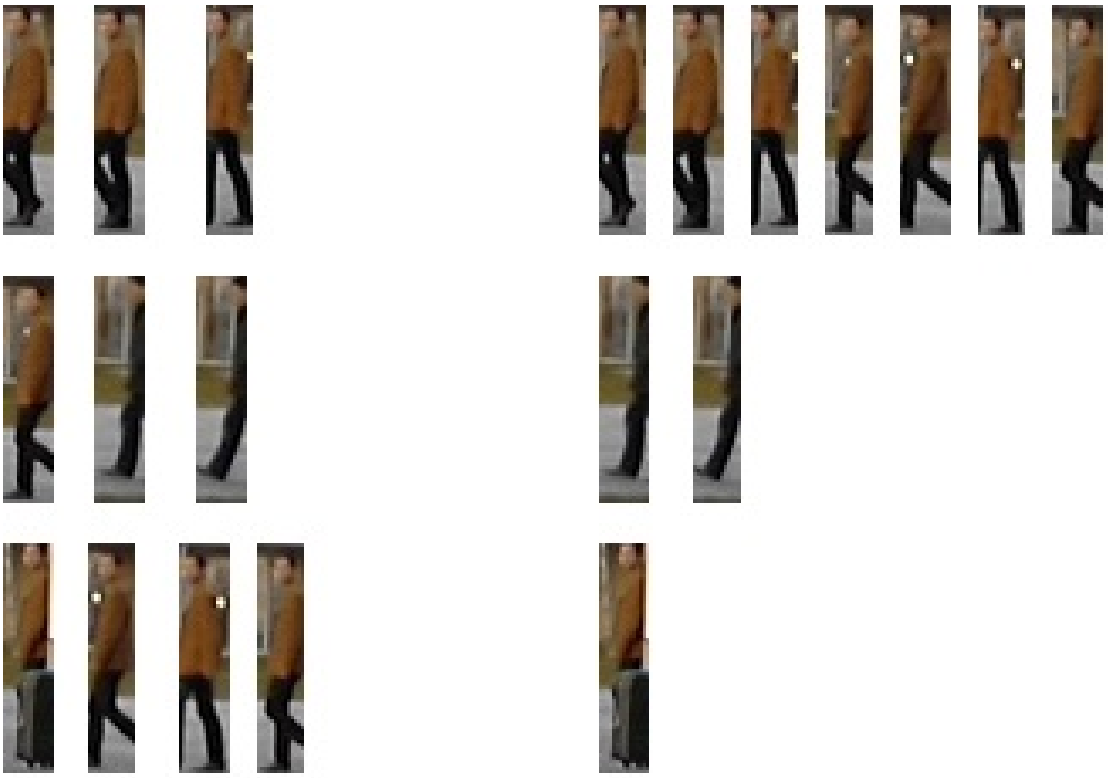


FIGURE 3.6: A clustering example: 10 samples on the left are simultaneously clustered into 3 groups (each row for one group) on the right according to their mutual similarities.

updating is going to take place. On the other extreme, setting γ to 1.0 makes the model totally "forgets" its appearance history.

The clustering-based appearance model updating procedure is summarized in Algorithm 2.

Algorithm 2 Clustering-based method for updating the object appearance model

Input: the most recent N collected samples,
the initial appearance model M_r^0,
the current appearance model M_r.

Output: the updated generative model M_r.

1: Obtain a number of clusters by performing the mean-shift clustering process in the feature space among the N samples.
2: Compute the sample mean of each cluster.
3: Find the mean $\bar{M}_s$ that has the smallest distance to M_r according to (3.18).
4: Update the object appearance model using M_r^0, $\bar{M}_s$ and M_r according to (3.21) and (3.22).

3.4.4 Evaluation of the tracking system

To evaluate the performance of the proposed tracking system, we compared it with some other tracking methods on several challenging video sequences.

3.4.4.1 Experimental setup

The typical settings for each component of the proposed tracking system are as follows. The feature set for the adaptive covariance descriptor is the $f(x, y)$ in Equation (3.16). The number of retained principal components is fixed to 14. If the whole image frame is large, the algorithm searches in a local window as specified in the dynamical model (3.17) to accelerate the processing speed. The size of the local search window is set to be proportional to the size of the object. Otherwise, the algorithm searches in the whole frame. In both cases, the searching step is fixed to 4 pixels horizontally or vertically.

For updating the appearance model using the clustering-based method, the updating cycle is typically set to 10-15 frames. In general, longer cycles make the model less adaptive but more stable. Besides, longer cycles can enable the appearance model being tolerant to longer occlusions. However, if the appearance of the object changes quickly, long updating cycle may be retarded. On the contrary, shorter cycles keep the model better up-to-date but more prone to contamination of the model. Tradeoff between adaptivity and stability is to be considered. Setting the updating cycle to 10-15 frames can generally handle short-term tracking mistakes while keeping the model freshly adaptive to changes. In our experiments, the updating cycle was set to 10 frames. The bandwidth h of the mean-shift procedure is set to 1.5 for 10-dimensional vectors (after PCA dimension reduction). The linear combination coefficients α, β and γ in Equation (3.21) are indeed the learning the rates of the appearance model. Greater α is more conservative and pull the model towards the initial one M_r^0. Larger γ makes the model adapts to changes quickly but also forgets its historical appearances quickly. Similar to the updating cycle, balance between stability and adaptivity is to be considered when choosing the mixture parameter for a specific application. A reference setting for α, β and γ is 0.10, 0.30 and 0.60 respectively. In our experiments, we used this reference setting for the "David-outdoor" sequence and the "White-outdoor" sequences because there are severe occlusions in these two sequences. In this case, both stability and adaptivity need to be considered. For the "Pedestrian1" sequence, we set α, β and γ to 0, 0, and 1.0 respectively, because there is no occlusion in this sequence and the appearance change is rapid.

Three benchmark sequences were used to assess the tracking performances. The first sequence is the "David-outdoor" sequence from [21], where the target undergoes partial occlusion, total occlusion, pose change and nonrigid deformation. The second one is from a self-captured video, where two pedestrians walk in in an outdoor campus environment, occasionally occluded by the background. This sequence contain occlusions, appearance variations and cluttered scenes. We refer to it as the "White-outdoor" sequence because the target pedestrian is in white clothes in the scene. The third sequence is the "Pedestrian1" sequence from the TLD dataset [6]. This sequence is captured by a moving camera, hence, with unpredictable camera movings and large appearance variations.

We compared the proposed method with a state-of-the-art tracking method, the "MILTracker" from [4], which uses online multiple instance boosting to handle partial occlusion. Moreover, to validate the effectiveness of the clustering-based model updating method, we explicitly compared it with two other updating schemes by keeping other components the same. For noting

convenience, the proposed tracking system that uses the adaptive covariance-based appearance model and the clustering-based model updating model is denoted as "AdpCov+cu". The first comparing updating scheme is to fully adapt to the changes at every frame using the mean of current tracking result and the last model updated as in [119]. We denote the tracking system using this updating policy as "AdpCov+fu". The other updating method is to use fixed initial model M_r^0 without updating. The corresponding tracking system is denoted as "AdpCov+nu".

The sequences are labeled with the "ground truth" for each frame. Percentage of Correctly tracked Frames (PCF) was employed to quantitatively measure the performance of all the involved trackers. PCF computes the percentage of correctly tracked frames over the total number of frames in the sequence. Tracking is considered to be correct if the overlap of the bounding box of the tracking result and that of the ground truth is greater than 25% of the area of the ground truth.

3.4.4.2 Results

The performances in terms of PCF of all the comparing tracking methods are presented in Table 3.2. The "AdpCov+cu" achieved the best results on all the three sequences. For qualitative comparison, we display a few screen snapshots of tracking results on the "David-outdoor" sequence, the "White-outdoor" sequence and the "Pedestrian1" sequence using all the comparing tracking methods in Figure 3.7, Figure 3.8 and Figure 3.9 respectively[4].

TABLE 3.2: Information of the sequences and the tracking performances in terms of PCF

Sequence	David-outdoor	White-outdoor	Pedestrian1
Number of frames	251	305	140
Frame size	640×480	640×480	320×240
Initial object size	38×126	16×70	16×65
Severe occlusion(s)	Twice	4 times	None
MILTracker	48.4 %	9.84%	69.78%
AdpCov+cu	96.0 %	93.44%	97.84%
AdpCov+nu	96.0 %	68.20%	61.15%
AdpCov+fu	12.8 %	6.89%	28.78%

The "MILTracker" generally fails when there are severe occlusions (see frame #100 in Figure 3.7) or fast appearance changes (see frame #113 in Figure 3.9). The "AdpCov+nu" tracker drifts to non-targets when the target undergoes significant appearance deformation (see Frame 200 and Frame 300 in Figure 3.8) or when there are similar non-targets in the scene (see Frame #113 in Figure 3.9). On the other hand, the "AdpCov+fu" tracker usually leads to tracking failure because its object appearance model is eventually contaminated due to updating during

[4]The tracking videos are available at:
https://www.youtube.com/watch?v=lB8y4D6jG7A
https://www.youtube.com/watch?v=N5DiqNSt4EA and
https://www.youtube.com/watch?v=d11dXuvxWKs.

FIGURE 3.7: Tracking results on the "David-outdoor" sequence using different updating policies. Column 1: "MILTracker". Column 2: "AdpCov+cu". Column 3: "AdpCov+nu". Column 4: "AdpCov+fu". In the last three columns, the blue rectangles indicate the search windows and the green rectangles show the tracking results. Row1: frame #1. Row 2: frame #100. Row3: frame #200. Row4: frame #250.

occlusion (see frame #100 in Figure 3.8) or accumulated tracking errors (see frame #50, #113 and #136 in Figure 3.9).

We see from the above results that the clustering-based updating method can effectively tolerate short-term tracking mistakes, including drifting to non-targets, partial/full occlusions, keeping the object appearance model steadily attached to the object and being up-to-date. The integrated tracking system "AdpCov+cu" accomplished stable tracking and outperformed a state-of-the-art tracker.

3.5 Conclusion

We have analyzed the generalization ability of the covariance descriptor and revealed that small eigenvalues may incur large variance and thus degrade its generalization. Generally, fusing more features can yield better detection performance, as long as it does not incur the small eigenvalue problem. When there are very small eigenvalues, the covariance descriptor hardly generalize. It is the logarithm function in the distance metrics that causes large disturbance and thus degrades the descriptor's generalization ability.

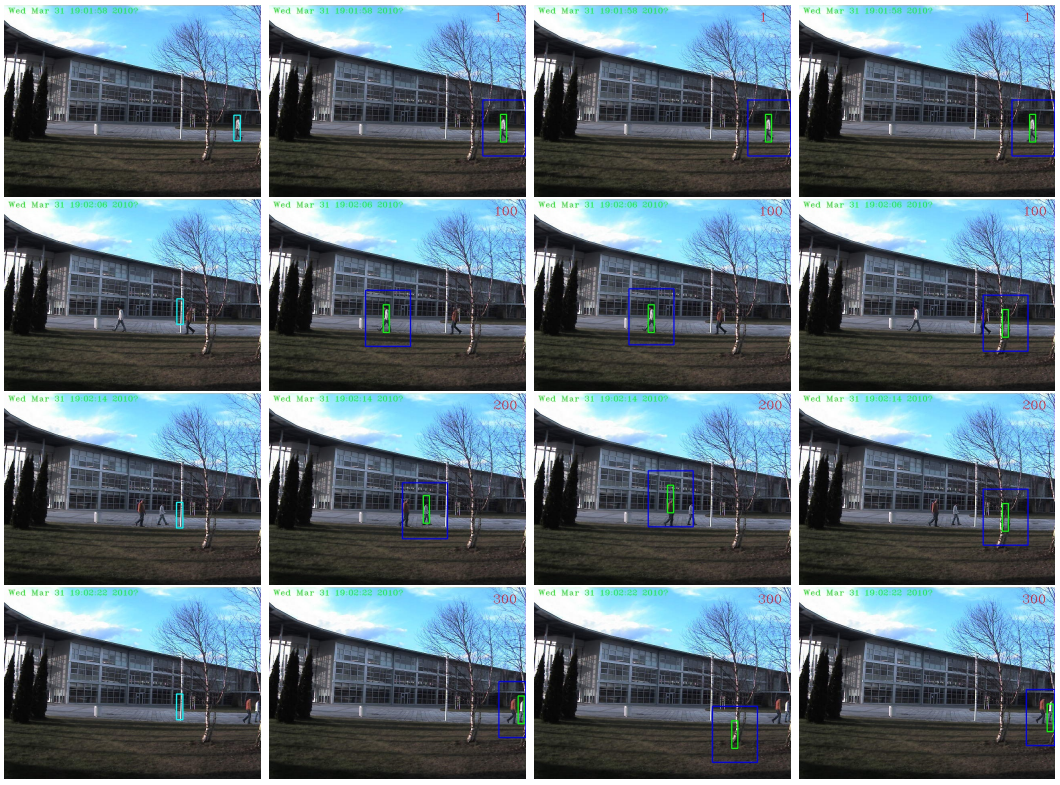


FIGURE 3.8: Tracking results on the "White-outdoor" sequence using different update policies. Column 1: "MILTracker". Column 2: "AdpCov+cu". Column 3: "AdpCov+nu". Column 4: "AdpCov+fu". In the last three columns, the blue rectangles indicate the search windows and the green rectangles are the tracking results. Row1: frame #1. Row 2: frame #100. Row3: frame #200. Row4: frame #300.

Regularization can effectively cure this problem and meanwhile preserve all the information in the corresponding dimensions. PCA dimension reduction, on the other hand, removes the unreliable dimensions, which can also be viewed as an adaptive feature extraction process. As dimensionality is reduced, operations on the adaptive covariance descriptor are generally less time-consuming.

The clustering-based updating method is able to select a group of reliable samples to update the object appearance model, which is particularly useful when there are tracking mistakes during tracking. Compared with other updating schemes, the clustering-based method showed merits in both adaptivity and stability.

The tracking system that integrates the adaptive covariance descriptor and the clustering-based updating accomplished stable tracking in several challenging real-world video sequences and outperformed a state-of-the-art tracker. We thus believe that the two components proposed in this work can be served as building blocks for constructing more robust tracking systems.

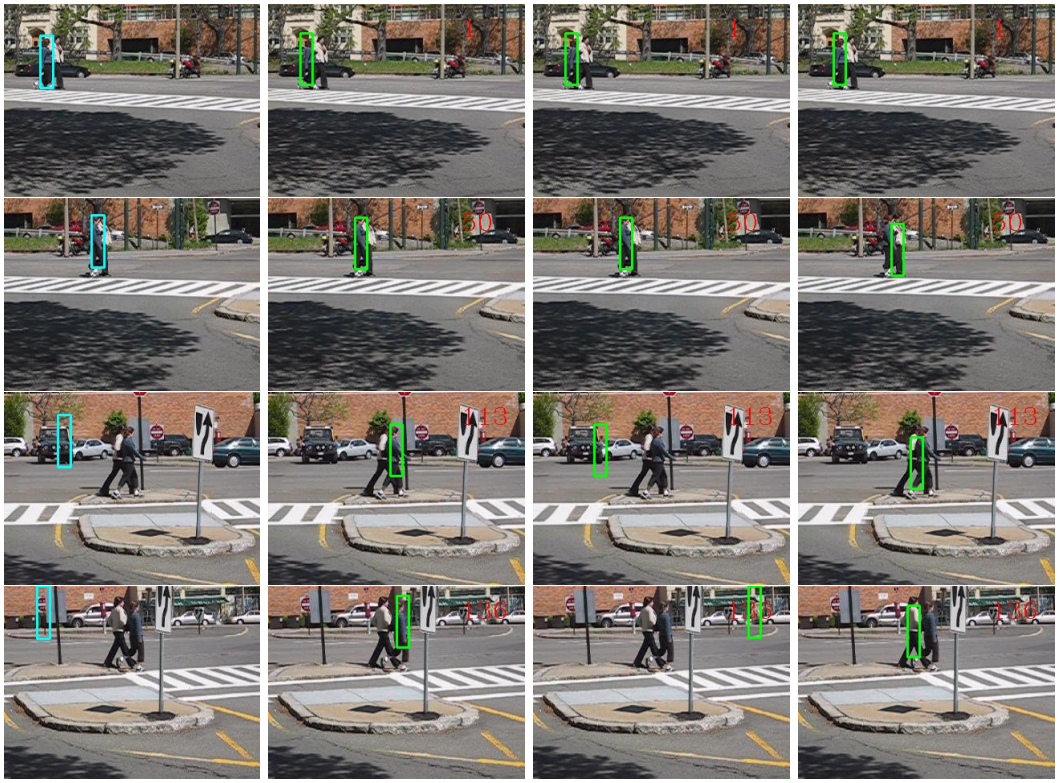


FIGURE 3.9: Tracking results on the "Pedestrian1" sequence using different update policies. Column 1: "MILTracker". Column 2: "AdpCov+cu". Column 3: "AdpCov+nu". Column 4: "AdpCov+fu". Row1: frame #1. Row2: frame #50. Row3: frame #113. Row4: frame #136.

Chapter 4

Online Learning Partial Least Squares Discriminant Model

4.1 Introduction

The tracking system developed in Chapter 3 has some limitations. First, it uses generative appearance model and did not exploit the background information. A common weakness of generative models is that they are prone to similar non-targets in the background, called "distracters", especially in cluttered scenes. Second, in cases where the target undergoes long time occlusion or absence from the scene, the object appearance model will inevitably be contaminated with non-target samples. To tackle these problems, we also resort to discriminative models for distinguishing the object from distracters and for preventing updating the model during target occlusion or absence. In particular, we use the Partial Least Squares (PLS) analysis to build discriminative object appearance models.

Recently, PLS-DA (Partial Least Squares Discriminant Analysis) has attracted increasing attentions in computer vision community. This may be attributed to its simplicity, effectiveness, efficiency and the fact that it has a unique parameter, which is intuitive and easy to tune. In fact, PLS-DA has been successfully applied to pedestrian detection [120], face identification [121, 122], object tracking [123] and discriminative appearance model learning [45].

Despite its increasing popularity, all the aforementioned applications employed batch PLS algorithms, e.g. NIPALS [124] or SIMPLS [125], which require maintaining all the training samples and retrain the PLS model, each time when some new training data is available. Due to their storage and computational requirements, these batch methods are unsatisfactory for real-world applications. First, they use the entire set of training samples for each update. If an updating is made at each time step, then the number of samples which must be retained grows linearly with the length of the time series. Second, the cost of computation grows with the number of samples, so they will run ever slower as time progresses.

To the best of our knowledge, few works have been proposed in the computer vision literature for incremental PLS model updating. This may be due to the iterative computation procedure of the PLS, which makes incremental methods not straightforward. This chapter is devoted to review PLS algorithms and to develop an online PLS-1 algorithms that can update the model incrementally or decrementally. Online learning discriminative object appearance model using the incremental PLS for visual tracking will be addressed in the next chapter.

In Section 4.2, we review classical PLS methods, the NIPALS algorithm and the SIMPLS algorithm. The proposed online PLS learning methods are developed in Section 4.3 based on an alternative non-iterative PLS solution. The proposed online PLS methods are evaluated in Section 4.4. Conclusions are drawn in Section 4.5.

4.2 The PLS analysis

The Partial Least Squares (PLS) regression is a statistical method which models relations between sets of observed variables X and Y by means of latent variables. It is a powerful statistical tool that consists of dimension reduction (compact feature extraction) and regression techniques, while considering the response variables in the process.

It constructs new predictor variables, known as components, as linear combinations of the original predictor variables, with consideration of the observed response values. According to whether Y is a vector or a matrix, PLS is categorized into two algorithms, the PLS-1 and PLS-2. Furthermore, by setting Y as categorical labels, PLS can be applied as a discriminant tool for the estimation of a low dimensional space that maximizes the separation between samples of different classes. This is the so called Partial least squares Discriminant Analysis (PLS-DA).

We shall review two popular PLS algorithms, namely NIPALS and SIMPLS, and point out the difficulties for developing online methods based on these algorithms.

4.2.1 The NIPALS Algorithm

Let $X \in \Re^{N \times r}$ be a mean-centered matrix of predictor variables, with rows corresponding to observations and columns to variables and $Y \in \Re^{N \times m}$ be the mean-centered response matrix. PLS methods find new spaces where most variations of the observed samples can be preserved, and the learned latent variables from two blocks are more correlated than those in the original spaces

$$\begin{aligned} X &= TP^\top + E \\ Y &= UQ^\top + F \end{aligned} \tag{4.1}$$

where $T \in \Re^{N \times p}$ and $U \in \Re^{N \times p}$ are factor (score, component, latent variable) matrices, $P \in \Re^{r \times p}$ and $Q \in \Re^{m \times p}$ are loading matrices, and $E \in \Re^{N \times r}$ and $F \in \Re^{N \times m}$ are error terms. Discriminative features T are extracted and the dimension is reduced when $p < r$.

To decompose X and Y by Equation (4.1), the nonlinear iterative partial least squares (NIPALS) algorithm [124], which is the classical form of the PLS method, performs in an iterative fashion. In the first iteration, the algorithm computes two weight vectors w_1 and c_1 such that most variations in X and Y can be retained in $t_1 = Xw_1$ and $u_1 = Yc_1$, while optimizing the covariance between the two score vectors as

$$\max [cov(t_1, u_1)]^2 = \max_{||w_1||=||c_1||=1} [cov(Xw_1, Yc_1)]^2 \tag{4.2}$$

where $cov(t_1, u_1) = t_1^\top u_1/N$ denotes the sample covariance between t_1 and u_1. The optimal weight vector w_1 and c_1 for the above optimization problem (4.2) are the first left singular vector and the first right singular vector of the cross scatter matrix $X^\top Y$ respectively [126]. Mathematically, the singular value decomposition (SVD) of $X^\top Y$ is written

$$X^\top Y = U\Lambda V^\top;\ U = [u_1\ \cdots\ u_r],\ V = [v_1\ \cdots\ v_m]. \tag{4.3}$$

w_1 and c_1 are thus obtained by setting

$$\begin{aligned} w_1 &= u_1, \\ c_1 &= v_1. \end{aligned} \tag{4.4}$$

When w_1 and c_1 are available, the score vectors t_1 and u_1 (first columns of T and U) can be computed by $t_1 = Xw_1$, $u_1 = Yc_1$, and loadings p_1 and q_1 (first columns of P and Q) can be computed by $p_1 = \frac{X^\top t_1}{t_1^\top t_1}$ and $q_1 = \frac{Y^\top u_1}{u_1^\top u_1}$. The data matrices X and Y are then deflated by subtracting their rank-one approximations

$$\begin{aligned} X &\leftarrow X - t_1 p_1^\top \\ Y &\leftarrow Y - u_1 q_1^\top. \end{aligned} \tag{4.5}$$

After the first step, the deflated X and Y are used to compute w_2 and c_2 based on Equation (4.3) and (4.4). This process is repeated iteratively until the residuals are small enough or a predefined number of weight vectors $w_1, \ldots, w_p$ are obtained. Such deflation rule ensures orthogonality among the latent vectors t_i and also among the weight vectors w_i that are extracted over the iterations.

For PLS regression, a linear relation between the score vectors t and u exists; i.e.

$$U = TD + H \tag{4.6}$$

where $D \in \Re^{p \times p}$ is a diagonal matrix and H denotes the matrix of residuals. It follows that Y is regressed by T as

$$Y = TDQ^\top + (HQ^\top + F). \tag{4.7}$$

By integrating the relationship [127]

$$T = XW(P^\top W)^{-1} \tag{4.8}$$

where P is the loading matrix defined in Equation (4.1), the overall (from X to Y) regression equation is written

$$Y = X(W(P^\top W)^{-1}DQ^\top) + F^*, \tag{4.9}$$

where $F^* = HQ^\top + F$ is the overall residual. The overall regression coefficient β is thus formulated as

$$\beta = W(P^\top W)^{-1}DQ^\top. \tag{4.10}$$

For a test feature vector x_t, its regression response y_t is therefore evaluated by

$$y_t = (x_t - \mu(X))^\top \beta + \mu(Y), \tag{4.11}$$

where $\mu(X)$ and $\mu(Y)$ are the sample means of X and Y before the mean centering respectively. Additional details regarding PLS methods can be found in [128].

4.2.2 The SIMPLS Algorithm

For make this thesis self-contained, we present here another popular PLS method, the SIMPLS algorithm, as proposed in [125]. SIMPLS is generally faster than NIPALS because it does not deflate the X. Instead, it deflate the matrix $X^\top Y$, which is usually much smaller in dimension than X. The complete procedure for computing PLS using SIMPLS is summarized in Algorithm 3.

4.2.3 Discussion

We see that both algorithms require the raw data blocks, X or Y, to participate in the computation at each iteration. This dependency makes it difficult to update the model when new data is available in a sequential scheme. In addition, when the number of samples is large, these kinds of methods are rather time-consuming.

For online updating a PLS model, the conventional PLS algorithms are not appropriate. For each updating, the batch PLS algorithms would require to recompute the PLS model using the accumulated samples ever seen. For an object tracking application, the storage and computational requirements of the batch algorithms are ever increasing as tracking evolves.

4.3 Online PLS-1 methods

We limit our discussion to regression problems with a single dependent variable, i.e. the PLS-1 algorithm, and propose a novel PLS-1 learning algorithm which can update a PLS-1 model incrementally or decrementally.

Algorithm 3 SIMPLS algorithm for computing PLS model

Input: X: the independent data block
Y: the response data block
p: the number of factors (latent variable, retained components)

$Y_0 = Y - MEAN(Y)$
$S = X^\top \times Y_0$
for $i = 1 \cdots p$ **do**
  $q =$ dominant eigenvector of $S^\top \times S$
  $\vec{r} = S \times q$
  $t = X \times \vec{r}$
  $t = t - MEAN(t)$
  $normt = SQRT(t^\top \times t)$
  $t = t/normt$
  $\vec{r} = \vec{r}/normt$
  $p = X^\top \times t$
  $q = Y_0^\top \times t$
  $u = Y_0^\top \times q$
  $v = p$
  if $i > 1$ **then**
    $v = v - V \times (V^\top \times p)$
    $u = u - T \times (T^\top \times u)$
  end if
  $v = v/SQRT(v^\top \times v)$
  $S = S - v \times (v^\top \times S)$
  Store $\vec{r}$, t, p, q, u and v into R, T, P, Q, U and V, respectively.
end for
$\beta = R \times Q^\top$

Output: β

4.3.1 A closed-form PLS-1 solution

Rather than the conventional PLS algorithms, we adopt an alternative approach [129], which provides a closed-form PLS-1 solution. The closed-form PLS-1 solution takes two scatter matrices, namely S_{xx} and S_{xy}, as input to compute the PLS model instead of using the raw data blocks X and Y. The two scatter matrices are defined as

$$S_{xx} = \sum_{i=1}^{N} (x_i - \mu(X))(x_i - \mu(X))^\top \tag{4.12}$$

$$S_{xy} = \sum_{i=1}^{N} (x_i - \mu(X))(y_i - \mu(Y))^\top, \tag{4.13}$$

where N is the number of samples in X (and also Y) and $\mu(X)$ and $\mu(Y)$ are sample means of X and Y respectively. Note that in (4.12) and (4.13), each x_i and y_i are arranged in vector form and we have $S_{xx} \in \Re^{r \times r}$ and $S_{xy} \in \Re^{r \times m}$.

With S_{xx} and S_{xy}, the Krylov Matrix $K_r \in \Re^{r \times rm}$ of the pair (S_{xx}, S_{xy}) is defined as

$$K_r = [S_{xy} \quad S_{xx}S_{xy} \quad S_{xx}^2 S_{xy} \quad \cdots \quad S_{xx}^{r-1} S_{xy}]. \tag{4.14}$$

A reduced Krylov matrix $K_p \in \Re^{r \times pm}$ is formed by the first p ($1 \leq p \leq r$) columns of K_r:

$$K_p = [S_{xy} \quad S_{xx}S_{xy} \quad S_{xx}^2 S_{xy} \quad \cdots \quad S_{xx}^{p-1} S_{xy}]. \tag{4.15}$$

According to [129], the relationship between the weight matrix W of the trained PLS model and the Krylov matrix of the pair (S_{xx}, S_{xy}) is well established. It is revealed that for univariate Y, i.e. when $m = 1$, the conventional orthonormal weighting matrix $W_p \in \Re^{r \times p}$ using p latent variables and the Krylov matrix K_p span the same column space. Moreover, W_p can be computed directly by performing the QR decomposition (and take the Q part) or the (modified) Gram-Schmidt procedure on K_p.

Furthermore, the regression coefficient β can be computed in a direct formula either using K_p as

$$\beta_{K_p} = K_p (K_p^\top S_{xx} K_p)^{-1} K_p^\top S_{xy} \tag{4.16}$$

or using W_p as

$$\beta_{W_p} = W_p (W_a^\top S_{xx} W_p)^{-1} W_p^\top S_{xy}. \tag{4.17}$$

The two expressions β_{K_p} and β_{W_p} yield identical results because K_p and W_p span the same column space [129]. They correspond to the partial least squares (PLS) regression using p latent variables when $p < r$ and reduce to the ordinary least squares (OLS) regression (assuming that $K_r^\top S_{xx} K_r$ is nonsingular) when $p = r$.

In practice, the explicitly formulated Krylov matrix K_p in Equation (4.15) may be ill-conditioned due to accumulated round off errors when computing the powers of S_{xx}, especially when p is large. This adversely affects the accuracy of the resulted W_p and β. As suggested in [129], we use the Arnoldi's method [130] to extract the orthonormal basis of K_p from S_{xx} and S_{xy}. The pseudo code procedure of the Arnoldi's method for computing W_p is described in Algorithm 4, where $||\cdot||_F$ is the Frobenius norm. The regression coefficient β can thus be obtained using the resulting W_p according to Equation (4.17).

We remind that the above introduced non-iterative PLS solution works for univariate Y only, i.e. Y is vector instead of matrix, which is the case for many applications, e.g. [45, 120, 121, 123]. For detailed proof and further information of the non-iterative PLS solution, we refer the readers to [129].

It is worth noting that the two scatter matrices S_{xx} and S_{xy} are constant in size (independent of N) and can be updated incrementally with new samples, an incremental PLS model updating algorithm can thus be developed. In fact, the output W and β of a PLS model trained from the data blocks X and Y can be fully determined by S_{xx}, S_{xy} and the dimension of retained latent variables p using Algorithm 4 and Equation (4.17) respectively. Besides, in order to update S_{xx} and S_{xy}, it is necessary to store the number of samples $N(X)$ and the samples means $\mu(X)$ and

Algorithm 4 Arnoldi's method for computing orthonormal weight matrix W_p

Input: S_{xx} and S_{xy}: the scatter matrices
p: the number of retained components
Output: the weight matrix W_p

$w_1 \leftarrow S_{xy}/||S_{xy}||_F$
for $i = 2 \cdots p$ **do**
  $w_i \leftarrow S_{xx} w_{i-1}$
  for $j = 1 \cdots i-1$ **do**
    $h_{j,i-1} \leftarrow w_j^\top w_i$
    $w_i \leftarrow w_i - h_{j,i-1} w_j$
  end for
  $h_{i,i-1} \leftarrow ||w_i||_F$
  $w_i \leftarrow \frac{w_i}{h_{i,i-1}}$
end for
$W_p = [w_1\ w_2\ \cdots\ w_p]$

$\mu(Y)$. Therefore, we suggest to specify a PLS model trained from the data block X and Y as

$$\Theta(X, Y, p) = (N(X), \mu(X), \mu(Y), S_{xx}, S_{xy}, W, \beta). \tag{4.18}$$

The advantage of adopting this model is that all the elements specified in the model are of constant size (independent of the number of training samples), which makes the model having a constant space complexity.

4.3.2 Incremental PLS model updating

We now propose a novel incremental PLS (IPLS) updating method. Suppose we have trained a PLS model with training set X_1 and Y_1 with dimension p_1. The model is thus denoted as $\Theta(X_1, Y_1, p_1)$. When new samples, i.e. feature vectors X_2 with their corresponding labels Y_2, are available, the incremental updating algorithm seeks to update the PLS model Θ with X_2 and Y_2 without resorting to the original training set X_1 and Y_1.

We describe in the following the updating of each element in the model. Firstly, the first five elements for $\Theta(X_2, Y_2, p_2)$ are computed as $N(X_2)$, $\mu(X_2)$, $\mu(Y_2)$, S_{xx2}, S_{xy2} respectively. Incremental updating of $N(X)$, $\mu(X)$ and $\mu(Y)$ is straightforward:

$$N(X) = N(X_1) + N(X_2); \tag{4.19}$$

$$\mu(X) = \frac{N(X_1)}{N(X)}\mu(X_1) + \frac{N(X_2)}{N(X)}\mu(X_2), \tag{4.20}$$

$$\mu(Y) = \frac{N(X_1)}{N(X)}\mu(Y_1) + \frac{N(X_2)}{N(X)}\mu(Y_2). \tag{4.21}$$

The scatter matrix S_{xx} can be updated using the following equation:

$$S_{xx} = S_{xx1} + S_{xx2} + \frac{N(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top. \tag{4.22}$$

Similarly, S_{xy} can also be updated as

$$S_{xy} = S_{xy1} + S_{xy2} + \frac{N(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(Y_1) - \mu(Y_2))^\top. \tag{4.23}$$

The weight matrix W can thus be updated using the newly updated S_{xx} and S_{xy} according to Algorithm 4. Finally, the regression coefficient β is updated by Equation (4.17). We note that although p can be different from both p_1 and p_2, no information is lost, since the number of samples, the means and scatter matrices have embedded all information needed to update the model.

4.3.3 Decremental PLS Model Updating

It is interesting to note that in some applications, one needs to remove (as opposed to update) some samples. Now we have trained a PLS model on the data set of X_1 and Y_1, we need to update the model after removing a training data block X_2 as well as its corresponding response Y_2. This is the decremental PLS (DPLS) model update problem.

This can be a straightforward extension of the incremental updating procedure. We compute the number of data, and their mean:

$$N(X) = N(X_1) - N(X_2) \tag{4.24}$$

$$\mu(X) = \frac{N(X_1)}{N(X)}\mu(X_1) - \frac{N(X_2)}{N(X)}\mu(X_2), \tag{4.25}$$

$$\mu(Y) = \frac{N(X_1)}{N(X)}\mu(Y_1) - \frac{N(X_2)}{N(X)}\mu(Y_2). \tag{4.26}$$

Then it is not difficult to prove that the scatter matrices S_{xx}, S_{xy} can be updated as

$$S_{xx} = S_{xx1} - S_{xx2} - \frac{N(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top. \tag{4.27}$$

Similarly, S_{xy} can also be updated as

$$S_{xy} = S_{xy1} - S_{xy2} - \frac{N(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(Y_1) - \mu(Y_2))^\top. \tag{4.28}$$

The weight matrix W and the regression coefficient β are then updated using Algorithm 4 and Equation (4.17) respectively using the updated S_{xx} and S_{xy}.

4.3.4 Weighted online PLS model updating

In some applications, it is interesting to give different weights to different training samples when updating the model. For example, in visual tracking, when the target undergoes the appearance changes, it is likely that recent observations will be more indicative of its appearance than more ancient ones. Therefore, it may be desirable to focus more on recently-acquired images and

down-weight the contribution of earlier observations. On the other hand, for semi supervised learning, a classifier is trained using labeled data, it exploits a set of unlabeled data to improve its accuracy. In this case, one may need to give smaller weights to the unlabeled samples.

To tackle this problem, we propose a weighted extension of the IPLS called weighted incremental PLS (WIPLS) model update method. The key idea is the concept of the "effective number" of a sample. By default, all observations have the same weight of 1.0. If a sample is assigned with a weight of 2.0, the result would be the same as if we had repeated this sample twice when counting the sample number, computing the means and the scatter matrices. On the other extreme, a point associated with a weight of 0 would make the result as if it had not been included in the computation at all. For WIPLS, we assign weights to the two training blocks with two scalar factors f_1 and f_2 when updating the model. The effective number of samples $N(X)$ and sample means $\mu(X), \mu(Y)$ are updated with the weight factor f_1 and f_2 as

$$N(X) = f_1 N(X_1) + f_2 N(X_2), \tag{4.29}$$

$$\mu(X) = \frac{f_1 N(X_1)}{N(X)}\mu(X_1) + \frac{f_2 N(X_2)}{N(X)}\mu(X_2), \tag{4.30}$$

$$\mu(Y) = \frac{f_1 N(X_1)}{N(X)}\mu(Y_1) + \frac{f_2 N(X_2)}{N(X)}\mu(Y_2). \tag{4.31}$$

The scatter matrix S_{xx} can be updated using the following equation:

$$S_{xx} = f_1 S_{xx1} + f_2 S_{xx2} + \frac{f_1 f_2 N(X_1) N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top. \tag{4.32}$$

The derivation of formula (4.32) is presented in Appendix A. Similarly, S_{xy} is updated with forgetting factor f as

$$S_{xy} = f_1 S_{xy1} + f_2 S_{xy2} + \frac{f_1 f_2 N(X_1) N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(Y_1) - \mu(Y_2))^\top. \tag{4.33}$$

Finally, the regression model W and β can be updated via Algorithm 4 and Equation (4.17) respectively using the newly updated S_{xx} and S_{xy}.

It is easy to observe that when $f_1 = f_2 = 1.0$, WIPLS is identical to IPLS. Similarly, WIPLS is reduced to DPLS when $f_1 = 1.0$ and $f_2 = -1.0$. This indicates that WIPSL is the general method for updating PLS model and both IPLS and DPIS are special cases of WIPLS. Besides, it is worth noting that when $0 < f_1 < 1.0$ and $f_2 = 1.0$, f_1 is the so-called the "forgetting factor" because it weights less (forgets) the previously trained samples.

4.3.5 Regression residual

An issue that has not been discussed is the regression residual. For our online PLS-1 methods, the raw data blocks X and Y are not maintained. We therefore propose to measure the regression residual using S_{xx} and S_{xy}. Specifically, we calculate the residual of regressed $S_{xy} = X^\top Y$ using

$S_{xx} = X^\top X$ and β, that is

$$\epsilon = ||X^\top Y - X^\top X\beta||_F = ||S_{xy} - S_{xx}\beta||_F. \tag{4.34}$$

The percentage of explained norm is then computed as $1 - \frac{||\epsilon||_F}{||S_{xy}||_F}$.

4.3.6 Time and space complexities

The space complexity of the proposed algorithm is $O(r^2)$, where r is the dimension of the predictors in X. This is in line with the size of the scatter matrix S_{xx}. For incremental or decremental updating, typically we have $N(X_2) \ll p \ll r$. Therefore, the time complexity is analyzed as follows: computing S_{xx2} requires $O(r^2)$ operations; computing W using Algorithm 4 takes $O(pr^2)$ operations and computing β using Equation (4.17) consumes $O(rp^2) + O(p^3)$ operations. As both complexities are independent of the number of training set $N(X)$, constant time and space complexities are achieved. In contrast, for batched PLS algorithms, the effect of $N(X)$ cannot be ignored. In that case, the space complexity would be $O(N(X)r)$ and the time complexity be $O(N(X)r^2) + O(N(X)rp)$, which are ever increasing.

In practice, there are situations where we do not need to compute W and β immediately after some new data are available. We can thus encode the information by updating only the first five elements, i.e. N, $\mu(X)$, $\mu(Y)$, S_{xx} and S_{xy}, instead of storing raw data. Computation of W and β are performed whenever necessary. This can be referred to as the "light updating" operation, which further reduces the computational cost. Note that in the "light updating" case, in order to enable testing an unknown sample, the original means, i.e. the means $\mu(X)$ and $\mu(Y)$ before updating, need to be preserved.

4.4 Experiments

4.4.1 UCI dataset

In order to validate the effectiveness of the incremental and decremental PLS model updating approaches proposed in the above section, we conducted an empirical study on benchmark data set from UCI Repository [131]. The relative location of CT slices on axial axis data set[1] is used. The data were retrieved from a set of 53500 CT images from 74 different patients (43 male, 31 female). Each CT slice is described by two histograms in polar space. The first histogram describes the location of bone structures in the image, the second the location of air inclusions inside of the body. Both histograms are concatenated to form the final feature vector. Bins that are outside of the image are marked with the value -0.25. The class variable (relative location of an image on the axial axis) was constructed by manually annotating up to 10 different distinct landmarks in each CT Volume with known location. The location of slices in between landmarks was interpolated.

[1] URL: https://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis

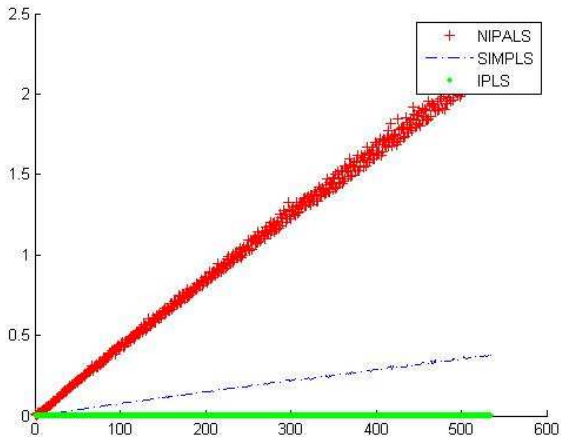


FIGURE 4.1: Computational time for NIPALS, SIMPLS and IPLS. X axis is the experimental step (535 in total) and Y axis is the computational time in seconds.

For training the PLS models, the X block is the feature vectors, i.e. concatenated histograms, and the Y block is the response data (class variables). We compared IPLS and DPLS with their batch counterparts. Without confusion, we denote PLS as the batch PLS methods in the following. For PLS, we employed the two most popular algorithms, NIPALS [124] and SIMPLS [125].

The following strategy is taken: the training sample is provided in an online way, with 100 new samples at each following step. At the initial step, both the PLS and the IPLS methods train a model using the initial 100 samples respectively. When new samples are available, PLS methods has to retrain the model and IPLS can update the model online according to the procedure described in section 4.3.2. As we have 53500 samples in total, PLS retrained the model 534 times and IPLS updated the model also 534 times. The number of retained latent variables P is set to 15 for all the three methods (NIPALS, SIMPLS and IPLS).

The experiments were carried out by running Matlab implementations on a desktop with 2.30GHz CPU and 12 GB memory. We recorded the Frobenius norm of the difference of the weight matrices W, and the Frobenius norm of the difference between regression coefficients β estimated by the two types of methods at each step. The computational time for (re)training or updating the models is also recorded.

Fig. 4.1 shows the computation time of the three methods each time when they retrain or update their models respectively. Not surprisingly, computation complexities of both NIPALS and SIMPLS grow linearly with the number of training samples. For NIPALS, average computational time is 1.1145 seconds and the value for SIMPLS is 0.1938 seconds. In contrast, computational time for IPLS is almost constant and average processing time is 0.0057 seconds.

Concerning accuracy, the average norm of differences between the weight matrices W produced by IPLS and that of NIPAL or SIMPLS[2] (we took a maximum) during the 534 updates is

[2]Actually, the weight matrix produced by SIMPLS is the R matrix in Algorithm 3, which is not the same as the W produced by NIPALS or IPLS. However, they share the same column space. Therefore, we first perform a QR decomposition of the R of SIMPLS and use the orthogonal basis Q for comparison.

$4.8131e^{-012}$, with a maximum value of $4.2417e^{-011}$. Likewise, the norm of the differences between regression coefficients β have an average value of $6.4392e^{-012}$ and a maximum value of $1.7628e^{-011}$.

When evaluating the DPLS method, we re-ran the experiments in a reverse way. We began with 53500 samples in the initial step and removed 100 samples at each step. Consequently, there were 534 times of retraining for NIPALS and SIMPLS and 534 times of updating for DPLS. For DPLS, the initial model was taken from the final model produced by IPLS in the last experiment.

As expected, our results showed that the computational time of NIPLS and SIMPLS in this setting decreased linearly with the number of training samples. Average processing time for NIPALS is 1.1053 seconds. It is 0.1918 seconds for SIMPLS and 0.0056 seconds for DPLS. The average norm of differences of W between DPLS and NIPALS or SIMPLS (the larger one is taken) is $1.2621e^{-009}$. The maximum norm of differences is $5.3754e^{-007}$. Average norm of differences of β is $7.2808e^{-010}$ with a maximum value of $2.1860e^{-007}$.

We see from the above results that the proposed IPLS and DPLS methods are both accurate and efficient. In terms of accuracy, the differences is still negligible after thousands of times of updating. On the other hand, substantial time gain was achieved using IPLS or DPLS. Although we didn't explicitly measure the space complexity, it is easy to see that the proposed IPLS and DPLS methods have constant space complexity.

4.4.2 VIPeR dataset

In this section, we train discriminative object appearance model using PLS and evaluate the performance of the model on real image dataset. The viewpoint invariant pedestrian recognition (VIPeR) dataset[3] of [132] is used as a benchmark. The VIPeR dataset consists of 632 pedestrian image pairs with large viewpoint, pose and lighting differences captured from two cameras. The objective is to recognize the corresponding image among a large number of pedestrians when provided with one image from the other camera. Some example images are shown in Figure 4.2. As the whole dataset is large, we used a subset consisting of the first 50 pedestrians.

In order to design a pedestrian recognition system, we use the first elements of images pairs (first view = first row in Figure 4.2) as training samples and the second elements (second view = second row in Figure 4.2) for testing the performances. We built discriminative object appearance models using PLS-DA. The features (the data block X in PLS) are covariance descriptors that are transformed into vector form as described in Chapter 3 Section 3.4.1. For each pedestrian, a PLS model is constructed by labeling the image as $y = +1$ for the pedestrian of interest and $y = -1$ for the remaining pedestrians. The number of retained variables of PLS are set to 10. After training, a ranking is then performed by sorting the PLS regression output responses of the images of the second view.

For comparison of performance, we use generative models associated with distance measures as described in Chapter 3. Then, during testing, a ranking is obtained by sorting the distances

[3] Available at http://vision.soe.ucsc.edu/node/178

FIGURE 4.2: Some examples from the VIPeR dataset. Each column is one of 632 same-person example pairs. There are wide range of viewpoint, pose, and illumination changes.

between the descriptors of the candidate pedestrian from the second view and the pedestrian of interest in the first view. Both ℓ_1 and ℓ_2 norms were employed. We denote the generative model using ℓ_1 norm as GM1, the one using ℓ_2 norm as GM2. The discriminative model using PLS is denoted as as DM.

It is worth noting that a direct implementation of the above system requires the implementation of N PLS batch algorithms, where N is the number of pedestrians. However, we can also note that all these PLS share the majority of training samples: the difference between 2 PLS is only the labels of 2 pedestrians (the labeled dataset of the second PLS is obtained by just relabeling 2 samples). In this case, the proposed incremental and decremental PLS algorithm will be of great help as the training of one PLS could be based on a previous one by applying one step of decremental PLS (taking off 2 samples) and one step of incremental PLS (adding 2 samples). This implementation makes the adaptivity of the learning system very fast for large datasets of pedestrian.

We computed the recognition accuracies in terms of the cumulative matching characteristic (CMC) [132]. Figure 4.3 shows the CMC curve demonstrating the recognition performance of each model.

We display in Figure 4.4 some examples of pedestrians observed by a camera, and the corresponding most similar pedestrians found by the discriminative appearance model (DM) within another camera. The images are sorted (from left to right) and the the correct match is presented at the end.

We see that the discriminative model using PLS outperforms the generative ones by a considerable margin. This is enabled by the fact that the discriminative models can explore information from both positive and negative samples while their generative counterparts learn from only one

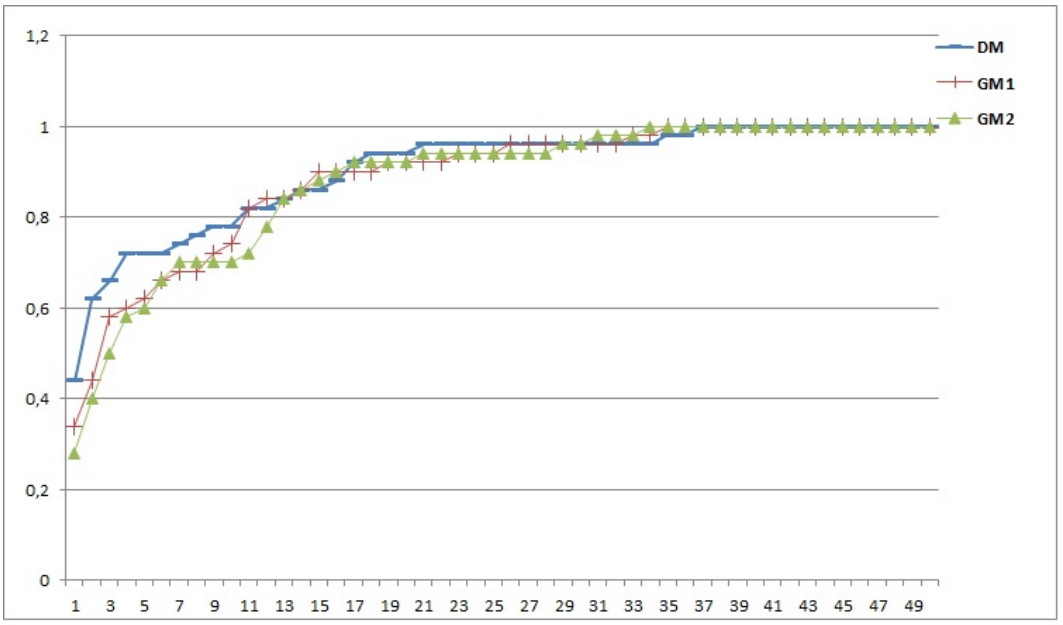


FIGURE 4.3: CMC curve of recognition performance using each appearance model. DM1 is the discriminative model using PLS. GM1 is the generative model using ℓ_1 norm. GM2 is the generative model using ℓ_2 norm.

FIGURE 4.4: Some example queries to recognition database using the discriminative appearance model trained by PLS-DA. Left column: the probe images. Middle columns: top 9 results sorted from left to right. Right column: the correct matches.

positive sample. This suggests that properly trained discriminative model is helpful for enhancing performance of recognition, re-acquisition, and tracking. Besides, we see that GM1 performs slightly better than GM2, which coincides with our analysis in Chapter 3 indicating that the ℓ_1 norm is generally more robust than the ℓ_2 norm.

4.5 Conclusion

We have presented online methods for updating PLS-1 regression models in an incremental or decremental fashion. The proposed methods have constant space and time complexities. It is observed that these incremental and decremental model update methods are special cases of a generalized weighted extension, which can assign weights to different training data blocks when updating the model.

The proposed incremental PLS-1 algorithm obeys the following general criteria for an incremental learning algorithm: 1)it does not require access to the original data; 2)it preserves previously acquired knowledge; 3)it is able to learn new information from new data. Analysis reveals that the proposed online updating algorithms possess the appealing property of constant storage and computational complexities while being accurate compared to their batch counterparts.

The proposed online PLS-1 algorithms are mathematically accurate. Deviations from batch methods may be observed due to machine precision and accumulated round off errors. Our experiments on UCI dataset showed that after thousands of updates, the observed deviation is still negligible. We therefore expect that the online PLS-1 approaches presented in this chapter can find applications in a variety of areas outside of visual tracking.

Our experiments on the VIPeR dataset showed that compared to generative models, training discriminative appearance models using PLS-DA is an effective way to improve recognition accuracy. In the next chapter, we will apply the PLS methods developed in this chapter for visual tracking.

Chapter 5

Cascaded Generative and Discriminative Object Appearance Models for Tracking

5.1 Introduction

In long-term unconstrained environments, motion is not a reliable cue. In fact, the target can be fully occluded or can leave the field-of-view for a long time. A robust tracker requires building a reliable appearance model used to efficiently reacquire the target and to continue its tracking when it reappears. Inspired by the cascaded face detector of [133], we propose a principled scheme fusing the merits of generative and discriminative models by incorporating them in a cascaded fashion: the generative model eliminates "easy" negative examples in the early layers of the cascade, while in the later layers, the discriminative model generates a decision boundary distinguishing the object from its most similar distracters. In particular, we employ a simple histogram-based generative model to filter out most easy candidate regions and retain a few prominent non-overlapping candidates. These retained samples are further re-evaluated by the discriminative model using the Partial Least Squares (PLS) discriminant analysis, which is able to distinguish the subtle differences between the target and its most confusing distracters. Both models are collaboratively updated online to adapt to appearance variations of the target and the background.

The first part of this chapter consists in proposing a cascaded framework that integrates both generative and discriminative appearance models. The cascaded appearance models for tracking is presented in Section 5.3. Our motivation for blending generative and discriminative models in such a cascaded manner can be explained in three folds. First, the cascaded structure can effectively tackle the asymmetry problem of training and testing data. Object tracking (and also object detection) has the intrinsic problem of unbalanced samples, i.e. the limited target instances are positive samples while all "the rest of world" being negative samples. In this sense,

the cascaded structure provides a remedy for handling the asymmetry problem. Second, the cascaded structure is more efficient, i.e. with lower computational complexity, than parallel configurations. Hybrid approaches that combine multiple models in parallel, e.g. using co-training [96], generally have more computational complexity because all candidate samples need to be evaluated by each model before making a final decision. Besides, these approaches still face the problem of unbalance data, which may adversely affect the performance of a discriminative classifier. Third, the underlying assumption of machine learning is that the training samples and the test samples are drawn from the same (although unknown) distribution. In the literature there has been a convention to randomly select samples for training or updating a discriminative appearance model. We argue that training and testing on pre-selected samples by the precedent layer in a cascaded structure is more likely to satisfy the assumption than the random selection policy. Therefore, better detection accuracy may be expected within a cascaded detection structure. The advantages of the cascaded detection structure has been evidenced by the success of the cascaded face detection system [133].

The second part of this chapter is the online updating of the appearance models in the detection cascade. Efficient and effective model updating is the key component for appearance model-based tracking systems. In Section 5.4, we will employ the clustering-based updating method for the generative model and the online PLS-1 method for updating the discriminative model. Both model updating methods can be performed efficiently which makes our tracking system practically applicable.

In order to balance adaptivity and stability and to increase robustness as well, the third part of this chapter is to embed multiple homogenous generative and discriminative models that are coherently integrated in the cascade framework. Tradeoff between stability and adaptivity is accomplished by adopting distinct learning rate for each layer of the cascade. The enhancement that embeds multiple generative models and multiple discriminative models is presented in Section 5.5.

The fourth part is an implementation and empirical evaluation of the proposed tracking framework presented in Section 5.6. We first diagnostically evaluated the contributions of the components in the system and then compared the overall system with the state-of-the-art. Diagnostic results suggest that cascade of generative and discriminative models is an effective fashion to boost detection performance and embedding multiple models is important for further accuracy improvement. Comparative results on challenging public video sequences showed superior or comparable performances with respect to the state-of-the-art.

5.2 System overview

Given a set of observed images $O_t = \{o_1, \cdots, o_t\}$, we aim to estimate the value of the hidden state variable Θ_t, which describes the affine motion parameters of the target at time t. In this work, we consider the state variable denoted by four affine transformation parameters as $\Theta_t = (x_t, y_t, s_t, \varphi_t)$, where x_t, y_t, s_t, φ_t denote x, y coordinates, scale ratio and aspect ratio

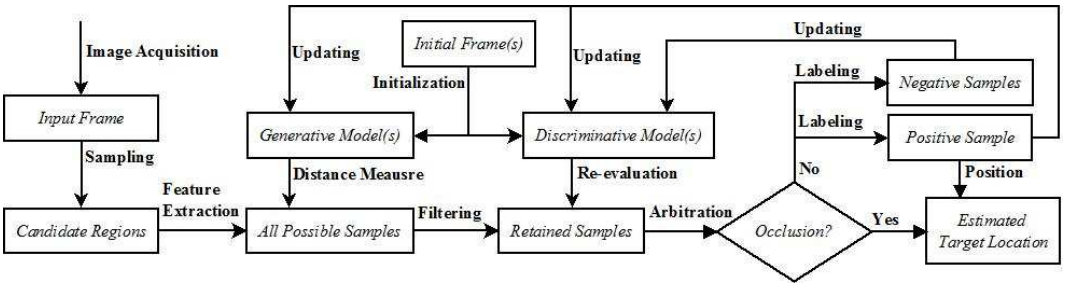


FIGURE 5.1: Overview of the cascaded tracking framework.

respectively. In practice, many candidate regions are sampled at every frame according to a Gaussian distribution around the lastly estimated target position:

$$p(\Theta_t|\hat{\Theta}_{t-1}) = N(\Theta_t; \hat{\Theta}_{t-1}, \Psi) \tag{5.1}$$

where Ψ is a diagonal covariance matrix whose elements are the corresponding variances of respective affine parameters, i.e., $\sigma_x^2, \sigma_y^2, \sigma_s^2, \sigma_\varphi^2$. Then, a maximum likelihood estimate $\hat{\Theta}_t$ could be approximated by locally searching among the sampled population as follows,

$$\hat{\Theta}_t = \arg\max_{\Theta_t} p(o_t|\Theta_t). \tag{5.2}$$

In the following, we focus on the observation model because it is crucial for determining the image region that is most likely to be the target object. As stated in the introduction section, we propose to use a cascade of generative and discriminative appearance models as our observation model for the likelihood measure. Roughly, the proposed observation model can be viewed as a cascade of two appearance models (one generative and the other discriminative) that work sequentially to boost the performance of the combined model. The intuition behind the cascaded tracker is that the generative model may make confusion between the target and similar background regions from time to time, it is safer to consider several most prominent candidates, as long as the real target is included in this collection. The discriminative model then specifically learns a decision boundary to distinguish the target from its rivals. In fact, the generative model provides the discriminative model with carefully selected training and testing samples while the discriminative tracker serves as performance re-evaluation or refining of the output of the generative model.

When designing the two appearance models, some considerations should be taken into account. For efficiency, the generative model is expected to adopt a feature descriptor and a model that are simple to compute. In contrast, the discriminative model requires more informational features for ensuring accuracy, i.e. the features need to contain the information that can distinguish subtle differences between the real target and the "distracters" similar to it.

We present an overview of the tracking framework as depicted in Fig. 5.1. The details of the cascaded observation model will be illustrated in the next section. The tracking process works as follows. For an incoming test frame, a large number of candidates are sampled by a random transition model. Features are then extracted for each sample. Assuming that both the generative and the discriminative models have been trained, a sample is firstly evaluated by

the generative model usually by distance measurement or probability estimation. Only a few samples are selected for re-evaluation by the discriminative model. New samples are labeled and both appearance models are updated using the new samples. Finally, the location of the target is estimated based on the positive-labelled samples.

5.3 Cascaded generative and discriminative object appearance models

5.3.1 Sample selection via the generative appearance model

The likelihood of all the possible states sampled by the dynamical model is firstly measured by the generative model. The generative model is required to be not only effective but also simple to compute. In our implementation, we consider the feature vector obtained by concatenating 16-bin intensity histograms from a spatial pyramid of 4 levels as the region representation. Specifically, at each level L, the patch is divided into $L \times L$ cells resulting in a 480-dimensional feature vector. The evaluation is a typical template matching process by computing the distances between test sample features and the current generative model. We adopt the Chi-square distance in this work to measure the similarity between a test feature vector H_t and the generative model M_g, which is defined as

$$d(H_t, M_g) = \sum_i \frac{(H_t(i) - M_g(i))^2}{0.5(H_t(i) + M_g(i))}. \tag{5.3}$$

We note that other features and distance measurements can also be considered as long as they meet the requirements for the generative model as previously stated.

If only the generative model was used, we merely retain the best match that has the smallest distance to the generative model and consider it to be the target in the current frame. On the contrary, for our cascaded tracker, a few most promising samples are retained in this stage. Selecting these important samples is performed in an iterative manner. Denote S_a as the set of all possible samples, we first select in S_a the sample s^* that has smallest distance to the generative model and add it to the retained sample set S_r. Next, we remove from S_a all the samples that have region overlap with s^*. The two steps are repeated iteratively till the desired number of samples have been collected in S_r. It can be seen that samples retained in S_r are mutually non-overlapping. The reason for this non-overlapping selection policy is to get promising samples globally from different maximal peaks, avoiding being trapped to the neighborhood of only one local maxima. The selection process is summarized in Algorithm 5, where $f(s)$ is the feature vector (which is in our case the concatenated intensity histogram) extracted from sample s and $ROI(s)$ is the bounding box of the image patch.

With the above procedure, the large number of "easy" samples that have been discarded in this stage can be regarded as having $p(o_t/\Theta_t)$ equal to zero, while the preserved samples in S_r will be re-evaluated by the subsequent discriminative model.

Algorithm 5 Selection of important samples via the generative model

Input: $\{S_a, N, S_r, M_g\}$
$S_a = \{all\ possible\ samples\}$,
$N:$ *the desired number of samples*,
$S_r = \emptyset:$ *the retained sample set*
Output: $\{S_r\}$
1: **while** $|S_r| \neq N$ **do**
2: $s^* \leftarrow \arg\min_{s \in S_a} d(f(s), M_g)$
3: $S_r \leftarrow S_r \cup \{s^*\}$
4: $S_a \leftarrow S_a$ - $\{s \in S_a | ROI(s) \cap ROI(s^*) \neq 0\}$
5: **end while**

5.3.2 Discriminative re-evaluation

The samples retained in the previous step are re-evaluated by a discriminative model using the partial least squares (PLS) [128] discriminant analysis. Training the discriminative appearance model using PLS with the new descriptor is addressed in 5.3.2.1. Sample re-evaluation using the discriminative model is presented in 5.3.2.2.

5.3.2.1 Training the discriminative appearance Model

For our visual tracking application, the matrix X in the PLS formulation is the matrix formed by accumulating vector representations as computed in the previous section. A discriminative model can be trained by setting Y as binary labels. Training samples are taken from the set S_r of samples retained by the generative model layer. As tracking evolves, two sets are constructed: a positive sample set B_p and a negative sample set B_n. These two sets are initialized as empty sets and then filled online when applying the PLS regressor which assigns labels to samples in S_r.

For training the discriminative model, we use more sophisticated region descriptor instead of the simple intensity histogram to increase robustness. In fact, in this discriminative layer, this is possible as the number of retained samples has been greatly reduced and generating complex descriptors for these fewer samples becomes affordable. Specifically, the adaptive covariance descriptor, which is introduced in Chapter 2 Section 3.3.3, is exploited for training the discriminative model.

At the very initial phase of tracking where the object appearance variation is not intense, the samples are simply labeled by the generative tracker until the minimal number of samples to perform PLS is collected. That is, the sample with the smallest distance to the generative model is labeled as positive ($y = +1$) and be added to B_p, and the others in S_r are added to B_n with $y = -1$. The state of the positive sample is then considered to be the state of the target in that frame. We note that the duration of this initial phase can be very short. For example, if we retain 15 latent variables for the PLS model, at least the same amount of samples is required. Suppose 4 samples are collected at each frame to S_r, this initial phase takes only 4 frames.

The PLS training is performed by the the closed-form PLS-1 method described in the Chapter 4 Section 4.3.1. After training, the sample means $\mu(X)$, $\mu(Y)$ and the regression coefficient β are saved for later evaluating test samples. Scatter matrices S_{xx}, S_{xy} are saved as well for updating the model.

As time evolves, the model is punctually updated with newly labeled samples. Discriminative re-evaluation using the learned PLS model and the labeling of new samples are addressed in the next section. The model updating will be addressed in Section 5.4.

5.3.2.2 Re-evaluation via the discriminative Model

Once the discriminative model is learned, it is used at every frame to re-evaluate the representation matrix X_r of samples retained in S_r using (4.11) to obtain response scores Y_r. It is worth noting that this testing process is particularly fast because only a single dot product of the feature vector with the regression coefficient β is needed in (4.11) to obtain the response from the PLS regression model.

According to the labeling scheme, the best match $\hat{s} \in S_r$ is chosen to be the one associated with the greatest response value $\hat{y}$. A predefined constant $\tau \in [0, 1)$ is employed as threshold to measure the significance of $\hat{y}$. If $\hat{y}$ has value no less than τ, the state of $\hat{s}$, namely $\Theta(\hat{s})$, is considered to be the estimation of $\hat{\Theta}_t$ and is thus considered as the tracking result. $\hat{s}$ will be labeled as a new positive sample, i.e. with $y = +1$ and will be added to the positive sample buffer B_p, while the other samples in S_r being labeled as negative (with $y = -1$) and added to the negative sample buffer B_n for later updating the models. Otherwise, that is if $\hat{y}$ is less than τ, an **occlusion** (actually it may also be the out of the field-of-view case) event is declared, all the samples in S_r are discarded and the state of the target $\hat{\Theta}_t$ in this frame is estimated using its last state $\hat{\Theta}_{t-1}$ in the previous frame. Algorithm 6 summarizes the PLS re-evaluation procedure.

5.4 Collaborative online model updating

Both the generative and the discriminative models need to be updated as tracking evolves in order to adapt to the variations of the target and the background. Except the **occlusion** case, selected samples in S_r are labeled as stated in the previous section and are utilized for updating the appearance models.

5.4.1 Updating of the generative model

The generative model considers positive samples only. A baseline mean update can be served for model updating. In order to increase robustness, we adopt the clustering-based updating scheme as proposed in Chapter 3 Section 3.4.3. That is, instead of updating at every frame, the clustering-based updating works less frequently, e.g. every 10 frames. It collects the positive

Algorithm 6 Re-evaluation via the PLS discriminative model

Input: $\{S_r, \beta, \tau, \hat{\Theta}_{t-1}, B_p, B_n, \mu(X), \mu(Y)\}$
Output: $\{\hat{\Theta}_t, B_p, B_n\}$

for $i = 1 \cdots |S_r|$ **do**
  $X_r^{(i)} \leftarrow feature\ of\ the\ i^{th}\ sample\ s^{(i)}\ in\ S_r$
  $Y_r^{(i)} \leftarrow (X_r^{(i)} - \mu(X))^\top \beta + \mu(Y)$
end for
$\hat{y} \leftarrow \max_{i=1...|S_r|} Y_r^{(i)}$
$\hat{s} \leftarrow \arg\max_{\hat{s} \in S_r, i=1...|S_r|} Y_r^{(i)}$
if $\hat{y} \geq \tau$ **then**
  $\hat{\Theta}_t \leftarrow \Theta(\hat{s})$
  $B_p \leftarrow B_p \cup \hat{s}$
  $B_n \leftarrow B_n \cup S_r \backslash \hat{s}$
else
  declare **occlusion/absent**
  $\hat{\Theta}_t \leftarrow \hat{\Theta}_{t-1}$
end if

samples in this period, i.e. the most recent 10 samples in B_p, and performs a mean-shift clustering among them. Outliers are filtered out by retaining only the most similar cluster (whose mean has the smallest distance to the current generative model). The updated model $\hat{M}_g$ is then determined as linear combination of the initial model M_g^0 (model computed from the initial frame), the current model M_g and the mean of the samples in the retained cluster $\bar{M}_s$ defined as in Equation (3.21).

One can note that, in this way, a sample is used to update the generative model if it survives the sample selection by the generative model, the re-evaluation by the discriminative model and also the cluster selection by the mean-shift clustering. This helps protecting the model from being contaminated and hence increasing robustness.

5.4.2 Updating of the discriminative model

With the online PLS-1 model learning methods described in Chapter 4, we are able to update the discriminative model. Except the **"occlusion/absent"** case, the discriminative model is updated at each frame using the proposed incremental PLS updating algorithm (with a forgetting factor) and the newly labeled samples, which are collected in that frame as described in section 5.3.2.2.

One practical concern when updating the PLS model is how to dynamically choose the number of retained components for the PLS model. One way to tackle this is to make the number of components adaptive to the regression residual, i.e. $||Y - X\beta||_F$, where $||\cdot||_F$ is the Frobenius norm and assuming that X and Y have been mean centered. As the original data blocks X and Y are not maintained in our incremental PLS algorithm, we use S_{xx} and S_{xy} to compute the regression residual as described in Chapter 4 Section 4.3.5.

A threshold $\tau_2 \in (0, 1)$ is defined to measure the significance of the percentage of norm explained by the model. When the explained percentage is less than τ_2, the algorithm automatically augment the number of components by one. This ensures that dominant information is explained by the regression model. We note that as there is the forgetting factor, this auto-determined number may also reach equilibrium instead of ever increasing.

5.5 Extensions

To further increase robustness in long-term tracking, combining more appearance models seems to be promising. As the cascaded detection structure is scalable, we show in this section extensions of the tracking system by coherently embedding more homogenous generative and discriminative appearance models into the cascade structure.

5.5.1 Multiple generative models

The assumptions behind the cascaded framework is that the target object is always included in the retained sample set by the generative model, as long as it is present in the frame. This is highly possible if the number of the retained samples are fairly large. However, if the unique generative model failed to retain the good positive sample in the sample set, the tracking framework will definitely fail. This problem is likely to happen when the features of the generative model are not effective for the specific scene because no prefixed feature can work effectively in all scenes.

To tackle this problem, we suggest to use multiple generative models that work in parallel. Formally, we use n generative models, each of which retains a sample set, denoted as S_r^i for the i^{th} generative model. The retained sample set of the overall generative model, denoted as $S_{r,0}$, is thus the union of all these individual sets:

$$S_{r,0} = S_r^1 \cup S_r^2 \cup \cdots \cup S_r^n. \tag{5.4}$$

One way to extend the current system to have multiple generative models is to use histograms of different color channels. As the generative models are distinct, samples retained by different generative models can be quite complementary.

An issue raised by using multiple generative models is that the samples retained by different generative models no longer guarantee the property of mutual non-overlapping. We therefore need some modifications in line 10 of Algorithm 6 when labeling new negative samples. Specifically, we label the samples that do not overlap with $\hat{s}$ as negative samples. Those who overlaps with $\hat{s}$ are discarded. Therefore, in the case of using multiple generative models, line 10 in Algorithm 6 is modified as

$$B_n \leftarrow B_n \cup \{s \in S_{r,0} | ROI(s) \cap ROI(\hat{s}) = \emptyset\} \tag{5.5}$$

For the updating of multiple generative models, each model is updated independently by the clustering-based method using its corresponding feature.

5.5.2 Multiple discriminative models

The discriminative model can also be augmented by integrating more discriminative models. Instead of blending them in parallel, we propose to combine multiple discriminative models in a sequential manner. That is, to extend the cascade structure with more layers.

The layer extension procedure is similar to the basic two-layered case. When we intend to add a new layer at the end of the cascade, we retain a few most promising samples (instead of only one sample), from the precedent layer, and then re-evaluate them in this newly added layer. As the cascade structure is scalable, more layers can be integrated at the price of higher computational complexity. Tradeoff between accuracy and efficiency is to be considered when designing a specific application. The re-evaluation procedure by each discriminative model is similar as described in Algorithm 6, except that for discriminative models in inner layers a set of samples are retained. For the i^{th} discriminative model, it evaluates the sample set $S_{r,i-1}$ passed from precedent layer and retains a smaller sample set $S_{r,i}$. At the initial phase, all the discriminative models are identically initialized. After tracking each frame, the i^{th} model is updated using the newly labeled samples in $S_{r,i-1}$.

An advantage of integrating multiple discriminative layers is that the sample asymmetry problem (i.e. positive samples are much fewer than negative samples) is more alleviated in later layers. In addition, with multiple discriminative models, different thresholds and learning rates can be adopted for each layer to achieve balance between stability and adaptivity. In general, higher thresholds and lower learning rates should to assigned to later discriminative layers, because they deal with fewer and harder samples.

5.5.3 Illustration

We depict a detection cascade with n_1 generative models and n_2 discriminative models in Figure 5.2, where $S_{r,i}$, $1 \leq i \leq n_2$, denotes the sample set retained by the i^{th} discriminative model. $S_{r,0}$ is the samples retained by the generative models and S_{r,n_2} contains the only one sample that will be considered as the target in the current frame. Note that when re-evaluated by whichever of the discriminative models, if the highest response value is smaller than the pre-defined threshold of that model, the cascaded detection process stops and an **occlusion/absent** event is declared.

We present in the following a detection example by a system with two generative models (embedded in one layer) and two discriminative models. The dynamic model generates 1000 samples for a test frame. The two generative models filter out most easy samples and each retain a set of 10 samples, stored in S_r^1 and S_r^2 respectively. The overall number of samples in $S_{r,0}$ passed to the first discriminative model is thus reduced to 20 (assuming that there are no duplicate samples in S_r^1 and S_r^2). After re-evaluation, 5 most promising samples are preserved and are

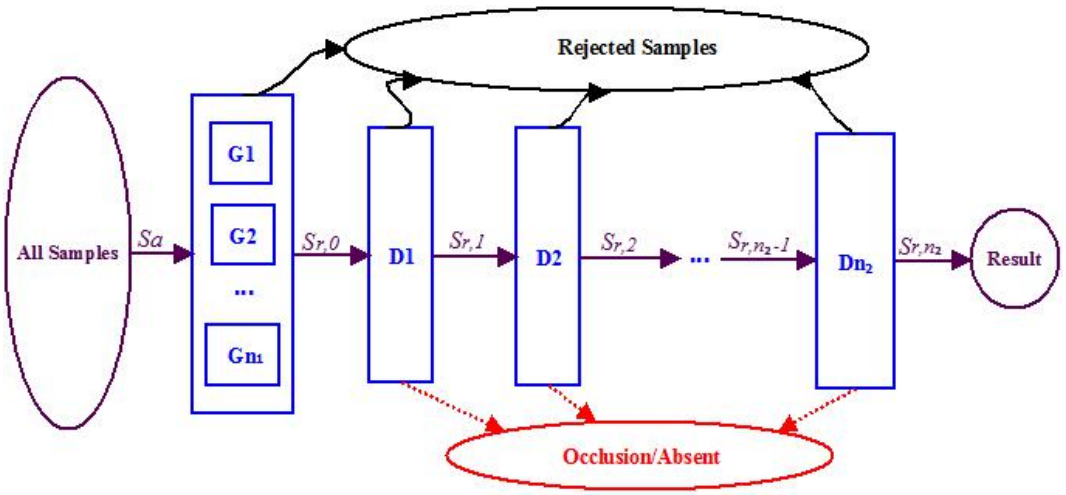


FIGURE 5.2: Cascaded detection structure with multiple generative models and multiple discriminative models. $G1, G2, \ldots, Gn_1$ are the $1^{st}, 2^{nd}, \ldots, {n_1}^{th}$ generative models respectively and $D1, D2, \ldots, Dn_2$ are the $1^{st}, 2^{nd}, \ldots, {n_2}^{th}$ discriminative models respectively.

passed to the second discriminative model, which then makes a final decision. A real detection scene is shown in Figure 5.3.

5.5.4 The overall tracking algorithm

We now summarize in Algorithm 7 the overall tracking algorithm described so far.

5.6 Experiments

We assessed the performance of the proposed tracking algorithm by conducting extensive experiments on challenging sequences and compared it with several state-of-the-art trackers.

5.6.1 Implementation details

Our current implementation of the proposed cascaded tracking system uses two generative appearance models and two discriminative models. As previously stated in section 5.5.1 and 5.5.2, the generative models work in parallel and the discriminative models work in sequential. The two generative models employ the histogram of pixel intensity and the histogram of the B color channel (from the RGB color space) as region descriptors respectively. The two discriminative models utilize the adaptive covariance descriptor (which is transformed into vector form as described in Chapter 3) in contrary.

For computing the adaptive covariance descriptor, we used a raw feature pool that contains a variety of features: pixel coordinates (x, y), color values from channels of a number of color spaces

Algorithm 7 The Overall Tracking Algorithm

Input: Image frames $F_1, \cdots, F_T$ and $\hat{\Theta}_1$
Output: $\{\hat{\Theta}_2, \cdots, \hat{\Theta}_t\}$

for $t = 1, \cdots, T$ **do**
  if $t = 1$ **then**
    Initialize the n_1 generative models and the n_2 discriminative models.
    Continue;
  end if

  if $t <= 2$ **then**
    Generate all possible samples by brute-force sliding window search.
  else
    Draw a number of candidate regions according to the dynamical model in (5.1) and compute the histogram features for each candidate region.
  end if

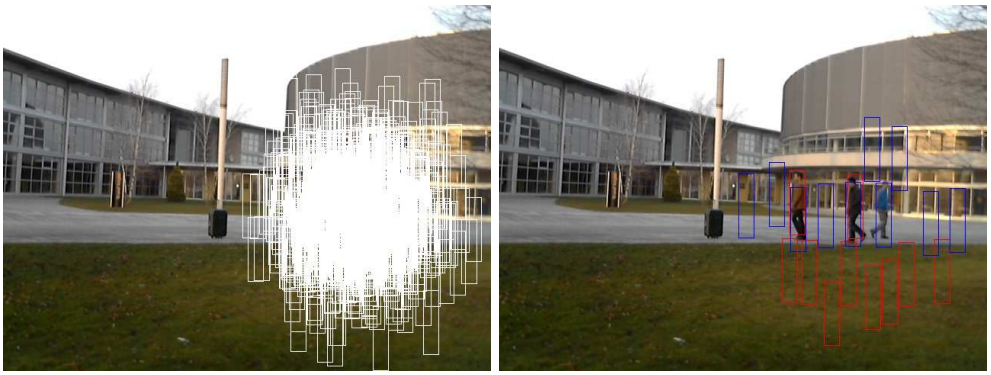
  for $i = 1, \cdots, n_1$ **do**
    Select a set of most important samples and store them in S_r^i using the i^{th} generative model according to Algorithm 5.
  end for

  Form the overall retained sample set $S_{r,0}$ by Equation (5.4) and compute the adaptive covariance descriptor-based representation for each sample in this set.

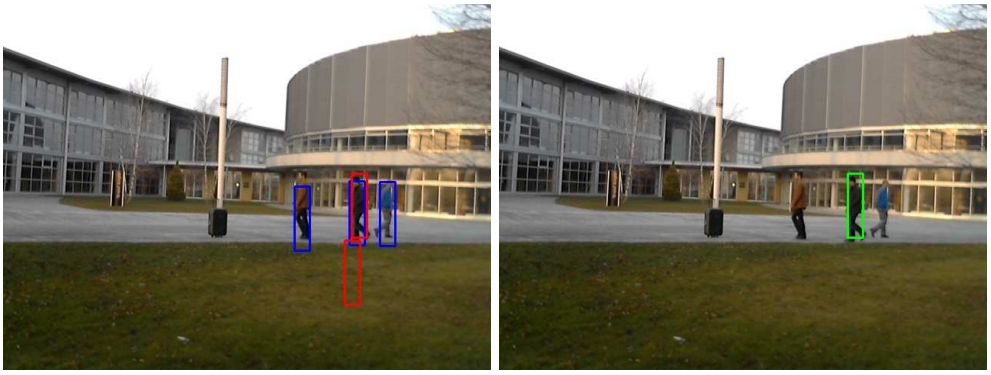
  for $j = 1, \cdots, n_2$ **do**
    Re-evaluate the samples in $S_{r,j-1}$ retained by the precedent layer via the j^{th} discriminative model according to Algorithm 6 and retain a most promising sample set $S_{r,j}$.
  end for

  if no **occlusion/absent** event **then**
    $\hat{\Theta}_t \leftarrow$ the state of the only sample in S_{r,n_2}.
    Update each discriminative model using the incremental PLS learning method with respective forgetting factor.
  else
    Declare that the target is **"invisible"**.
    $\hat{\Theta}_t \leftarrow \hat{\Theta}_{t-1}$.
  end if

  if the generative updating cycle is due **then**
    Update each generative model using the clustering-based method as in Algorithm 2.
  end if
end for



(a) (b)



(c) (d)

FIGURE 5.3: An example illustrating the output of each layer of the cascaded tracking framework which embeds two generative and two discriminative appearance models. In 5.3(a), the transitional model propose 1000 particles denoted by white rectangles; a set of 10 important samples is retained by each generative model in 5.3(b) denoted by **blue** or **red** rectangles according the generative model that has selected them; in 5.3(c) the selected 20 samples are re-evaluated by the first discriminative model and only 5 most promising candidates survive; finally the tracking result is produced after the re-evaluation of the 5 most promising samples by the second discriminative model and is shown in **green** rectangle in 5.3(d).

and the derivatives of the intensity image of various orders with respect to x and y. Specifically, each pixel in a region is converted to a 20-dimensional feature vector

$$
\begin{aligned}
f(x,y) = \Big[& x \;\; y \;\; R(x,y) \;\; G(x,y) \;\; B(x,y) \;\; H(x,y) \\
& L(x,y) \; S(x,y) \; a(x,y) \; b(x,y) \; u(x,y) \; v(x,y) \\
& \frac{\partial I(x,y)}{\partial x} \; \frac{\partial I(x,y)}{\partial y} \; \frac{\partial^2 I(x,y)}{\partial x^2} \; \frac{\partial^2 I(x,y)}{\partial y^2} \\
& \frac{\partial^2 I(x,y)}{\partial x \partial y} \; \frac{\partial^3 I(x,y)}{\partial x^2 \partial y} \; \frac{\partial^3 I(x,y)}{\partial x \partial y^2} \; \frac{\partial^4 I(x,y)}{\partial x^2 \partial y^2} \Big]^\top
\end{aligned}
$$

where x, y are the cartesian coordinates, R, G, B are the three channels from the RGB color spaces, H, L and S are the three feature channels from the HLS color space. Similarly a, b and u, v are the two color channels from the CIE Lab and CIE Luv color spaces respectively except the illumination channels L. We note that the coordinates and the color channels need to be adjusted to fall into the range of $[0-255]$. The intensity-image derivatives are computed as they

are using the Sobel operator with 3×3 or 5×5 kernels[1]. The number of retained eigenvectors for PCA projection is determined adaptively. Empirically, only those have corresponding eigenvalues greater or equal than 5.0 are kept.

For the dynamical model, 10000 particles are generated in each frame. The standard deviations of the affine transform parameters are set to $\sigma(x, y, s, \varphi) = [100\ 100\ 0.1\ 0.05]$. For generative appearance models, the updating cycle is set to 10 frames and the weights in Equation (3.21) for updating the model is set to $\alpha_1 = 0.15, \alpha_2 = 0.40$ and $\alpha_3 = 0.45$. Concerning the discriminative models, the number of retained samples by the first discriminative model, i.e. $|S_{r,1}|$, is set to 5. When initializing the discriminative models, the number of latent variables in the PLS models is set to explain at least 99.99% of the total information. This number is kept the same throughout the tracking. Thresholds for both discriminative models is set to be zero. Forgetting factor for the first discriminative model is 0.96 and that of the second layer is 0.999 (the higher the forgetting factor, the lower the learning rate). We kept these settings as the default values of the parameters for all the experiments unless stated otherwise.

Although the number of particles generated by the dynamical model is very large, we were able to compute the histogram feature vector of an arbitrary region for the generative models very quickly with the help of integral histogram [134]. In addition, our C++ implementation employed the Intel Threading Building Blocks (TBB) to take advantage of the speedup by multi-cores parallelization on modern CPUs. The tracking system runs at 8-15 frames per second (FPS) for 320×240 images on a 2.30 GHz CPU with 8 execution threads and 12 GB memory.

5.6.2 Diagnostics

In this subsection, we evaluate the contributions of the components in the cascaded tracking framework.

In order to validate the effectiveness of the proposed tracking framework in long-term unconstrained environment, we collected a series of challenging long-term sequences by ourselves. In the following, for notation convenience, we refer the proposed cascaded generative and discriminative tracker as CasGD tracker and the newly introduced sequences as the CasGD dataset.

The four sequences, namely View1, View2, View3 and View4, each consists of 640×480-pixels color images that are captured in a campus environment by cameras installed at different viewpoints. Each sequence contains two or three walking pedestrians with different color of clothes. We evaluated the performances of the CasGD tracker as well as the contributions of its components by trying to track each pedestrian in every sequence. When treated as tracking target, the pedestrians are denoted by the sequence name and the color of their clothes. For example, the pedestrian in blue in the second sequence is denoted as "View2-Blue". Similarly, "View4-Red" denotes the pedestrian in red to be tracked in the sequence View4. We therefore had 11 targets to track in total.

[1] According to the summarised order of partial derivatives, i.e. if the summarised order is less than 3, we use the 3×3 kernel; otherwise the 5×5 kernel is used.

The CasGD dataset is difficult because there are a combination of challenges in these sequences. The involved targets undergo significant changes in poses, scales, nonrigid deformations etc. In addition, there are abrupt motions, full occlusions and out of the field-of-views (absent), which are typical challenges for long-term visual tracking. Table 5.1 summarizes the challenges in the sequences when tracking each target. Fig. 5.4 shows snapshots corresponding to each sequence. To enable evaluating the tracking performance, the sequences were manually annotated. Those

TABLE 5.1: Challenges for tracking each target in the CasGD dataset.

Targets	Frames	Nonrigid deformation	Full occlusion	Partial occlusion	Out of view	Abrupt motion	Scale change
View1-Black	702	✔	✔	×	✔	×	×
View1-Blue	1058	✔	✔	✔	✔	×	×
View1-Yellow	713	✔	✔	×	✔	×	×
View2-Black	1131	✔	✔	×	✔	✔	✔
View2-Blue	80	✔	×	×	×	✔	✔
View2-Yellow	1049	✔	✔	×	✔	✔	✔
View3-Black	300	✔	×	×	×	×	✔
View3-Blue	415	✔	✔	×	×	×	✔
View3-Yellow	418	✔	✔	✔	×	×	✔
View4-Red	278	✔	✔	✔	×	×	×
View4-White	322	✔	✔	✔	×	×	×

(a) View1 (b) View2

(c) View3 (d) View4

FIGURE 5.4: Snapshots from the CasGD Pedestrian data set.

TABLE 5.2: Diagnostics of tracking performances in the CasGD dataset in terms of precision. The best performance is in **bold** and the second best is in *italic*.

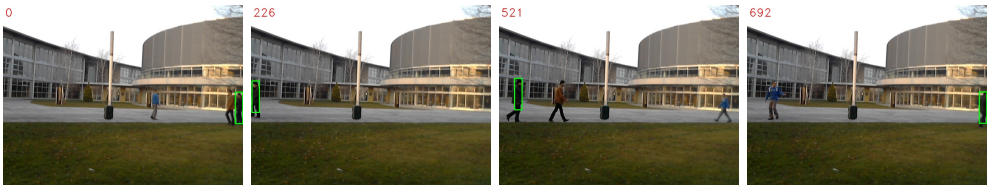
Target	T1	T2	T3	T4	CasGDT
View1-Black	32.81%	*97.72%*	40.14%	92.70%	**98.86%**
View1-Blue	18.34%	*99.34%*	18.92%	*99.34%*	**99.62%**
View1-Yellow	29.92%	**99.72%**	21.79%	*88.67%*	**99.72%**
View2-Black	11.24%	11.24%	*30.43%*	11.17%	**98.93%**
View2-Blue	93.67%	**100%**	**100%**	96.20%	**100%**
View2-Yellow	13.45%	*98.32%*	13.50%	92.65%	**99.90%**
View3-Black	99.00%	**100%**	63.57%	13.78%	**100%**
View3-Blue	93.48%	96.83%	*98.80%*	96.12%	**99.52%**
View3-Yellow	*62.50%*	61.82	37.07%	36.78%	**97.09%**
View4-Red	81.88%	**99.28%**	98.91%	98.55%	**99.28%**
view4-White	34.89%	*98.13%*	93.15%	91.32%	**98.44%**

with more than 75 percent of occlusion were annotated as "not visible".

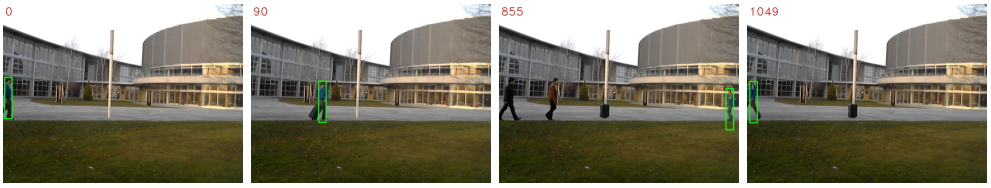
To attribute the contributions of the components, we compared the precisions of the overall CasGD tracker with a number of simplified variants. The first one, denoted as T1, used only one generative model (the intensity histogram) without discriminative models. The second one, T2, uses one generative model (the intensity histogram) and one discriminative model. To complete the comparison, we lack the configuration that uses one discriminative model without generative models. To implement it, we used the configuration with one generative model and one discriminative model. In contrast to T2, this third variant, denoted as T3, retains a large number of samples (50 samples in our implementation) after the generative selection. As such, the effect of sample selection by the generative model diminishes, making T3 close to the setting using one discriminative model and no generative model. The fourth variant, T4, employs two generative models (the intensity histogram and the blue histogram) and one discriminative model.

These four variants are compared with the overall tracking system, denoted as CasGDT, which uses two generative models (intensity and blue histograms) and two discriminative models. For faire comparison, all the other settings keep the same for all the involved trackers. Note that in the "View1" sequence and the "View4" sequence, we used fixed size without scale and aspect ratio change, i.e. $\sigma(s) = 0$ and $\sigma(\varphi) = 0$. In the "View2" and the "View3" sequences, we used the default setting, i.e. $\sigma(s) = 0.1$ and $\sigma(\varphi) = 0.05$.

Performances in terms of precision for tracking the 11 targets in the CasGD dataset are presented in Table 5.2. The overall CasGDT consistently achieved the best or one of the best results. This suggests that integrating multiple generative and discriminative models is important to increase tracking performance. We display in Figure 5.5 some representative snapshots when tracking several targets using the overall CasGDT tracker.



(a) View1-Black



(b) View1-Blue

(c) View2-Blue

(d) View2-Black

(e) View2-Yellow

(f) View3-Blue

(g) View3-Yellow

FIGURE 5.5: Some snapshots of tracking results using the CasGDT tracker on the self-captured CasGD sequences.

5.6.3 Comparison with the state-of-the-art

To better evaluate the performance of the proposed CasGD tracker, we also apply it on the challenging TLD dataset [6] and compared it with the results reported in [6]. The full TLD data set consists of ten sequences. Due to the color features used in CasGD, we took a subset of six sequences that consist of the color image, namely Pedestrian1, Pedestrian2, Pedestrian3, Motocross, Carchase and Panda. This set of sequences contains fast camera movings, total occlusions and dramatic target disappearances. In particular, the sequences Motocross, Carchase and Panda are long and contain all the typical challenges for long-term tracking. Specially, for the Pedestrian1, Pedestrian2 and Pedestrian3 sequences, σ_s and σ_φ are set to zero as they do not involve apparent scale or aspect ratio changes. For the Panda sequence, we used a larger standard deviation for aspect ratio by setting $\sigma_\varphi = 0.3$ because the nonrigid deformation of the target is intense in this sequence. Other settings use the default values specified in Section 5.6.1.

As in [6], the performance is evaluated using precision P, recall R and f-measure F. P is the number of true positives divided by number of all responses, R is the number true positives divided by the number of object occurrences that should have been detected. F combines these two measures as $F = 2PR/(P+R)$. We used the manually annotated ground truth provided by the authors of [6]. The ground truths were annotated such that frames where more than 50% of occlusion or more than 90 degrees of out-of-plane rotation were considered as "not visible". A detection was considered to be correct if its overlap with ground truth bounding box was larger than 25%. Those "not visible" frames were not counted during performance computation. This evaluation methodology is identical to that used in [6].

In [6], performances in terms of $P/R/F$ of seven trackers were reported. The seven trackers are the Online Boosting (OB) tracker [1], the Semi-supervised Boosting (SB) tracker [2], the Beyond Semi-supervised Boosting (BS) tracker [3], the Multiple Instance Learning (MIL) tracker [4], the Co-trained Generative and Discriminative (CoGD) tracker [5] and the Tracking Learning Detection (TLD) tracker [6]. TLD dominated in the comparison as it enabled re-detection of the object.

Table 5.3 summarizes the performances of all the 8 involving trackers. The last column shows the performance of CasGD and the results in the other columns are taken from those in [6]. CasGD achieved the best performance on 5 out 6 sequences. Compared to the TLD tracker, we examined the reasons for the inferior performances in sequence Carchase. In this sequence, low recall rate of the CasGD tracker is reported. This is due to the fact that the sequence is extremely long and it is very difficult (if not impossible) to distinguish the target only by its appearance in some part of the sequence. This problem occurs typically when the target, a car, becomes very small and situates among

TABLE 5.3: Tracking performances in the TLD dataset measured by Precision, Recall and F-measure. The best performance is in **bold**. CasGDT scored best in 5 out of 6 sequences. OB is in [1], SB in [2], BS in [3], MiL from [4], CoGD from [5], TLD from [6] and CasGDT is the cascaded generative and discriminative tracker presented in this work.

Sequence	Frames	OB	SB	BS	MIL	CoGD	TLD	CasGDT
1. Pedestrian1	140	0.61/0.14/0.23	0.48/0.33/0.39	0.29/0.10/0.15	0.69/0.69/0.69	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**
2. Pedestrain2	338	0.77/0.12/0.21	0.85/0.71/0.77	1.00/0.02/0.04	0.10/0.12/0.11	0.72/0.92/0.81	0.89/0.92/0.91	**0.97/0.97/0.97**
3. Pedestrian3	184	1.00/0.33/0.49	0.41/0.33/0.36	0.92/0.46/0.62	0.69/0.81/0.75	0.85/1.00/0.92	0.99/1.00/0.99	**0.99/1.00/1.00**
4. Motocross	2665	0.33/0.00/0.01	0.13/0.03/0.05	0.14/0.00/0.00	0.05/0.02/0.03	0.93/0.30/0.45	0.89/0.77/0.83	**0.87/0.88/0.88**
5. Carchase	**9928**	0.79/0.03/0.06	0.80/0.04/0.09	0.52/0.12/0.19	0.62/0.04/0.07	0.95/0.04/0.08	**0.86/0.70/0.77**	0.84/0.55/0.66
6. Panda	3000	0.95/0.35/0.51	1.00/0.17/0.29	0.99/0.17/0.30	0.36/0.40/0.38	0.12/0.12/0.12	0.58/0.63/0.60	**0.95/0.81/0.87**
mean	16255	0.74/0.09/0.14	0.72/0.08/0.14	0.56/0.11/0.18	0.47/0.12/0.13	0.79/0.13/0.18	0.8165/0.7091/0.7558	**0.8771/0.6759/0.7603**

a crowd of hundreds of similar cars on the road. The TLD tracker performed better in this case plausibly thanks to its embedded Median-Flow tracker [135] component which is extended with failure detection, and the P-expert component which verifies a reliable trajectory of the target. In other words, when the appearance of the target is not a reliable cue, the TLD tracker gain a distinct advantage entrained by exploring the temporal and spatial constraints during tracking. Nonetheless, the proposed CasGD achieved comparable or superior performances in the sequences where the appearance cue of the target is reliable.

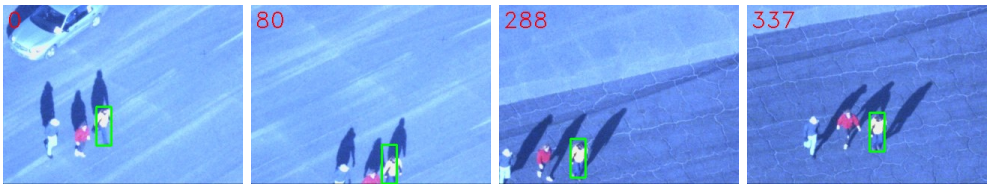
The last row of Table 5.3 shows a weighted average performance (weighted by number of frames in the sequence). Since the sequence Carchase contributes 9928 of total 16255 frames, very heavy weight is imposed by this particular sequence. The CasGD tracker scored the best with 76.03% in terms of the overall mean F-measure, which is slightly superior to that of TLD of 75.58%. Other approaches range between $13\% - 18\%$.

We display in Figure 5.6 some snapshots of tracking results using the proposed CasGDT tracker.

5.7 Conclusion

We have proposed a novel tracking method which combines the merits of both generative and discriminative models by incorporating them in a cascaded framework. The two types of appearance models work collaboratively to boost the performance. In particular, the generative model provides the discriminative tracker with carefully selected training and testing samples and the discriminative model explicitly learns the difference between the target and its most confusing counterparts. The proposed approach thus shows strong robustness against "drifting". In addition, the occlusion problem is also addressed and the proposed method can work in unconstrained circumstance where long time (full) occlusions or out of the field-of-view are present. Our empirical results

(a) pedestrain1



(b) pedestrian2

(c) pedestrian3

(d) motocross

(e) carchase

(f) panda

FIGURE 5.6: Some snapshots of tracking results using the CasGDT tracker on six sequences from the TLD dataset.

on challenging long-term sequences demonstrate that the cascade of generative and discriminative models is a good way to boost detection accuracy and embedding multiple models into the cascade structure can further improve robustness.

Chapter 6

Summary and Perspectives

6.1 Summary

Building a robust visual tracking system is a complicated task which requires designing multiple modules and structuring them in a coherent framework.

We have proposed novel approaches in several aspects and designed a robust visual tracking system. The final tracker developed in this thesis has the following innovative properties:

- region representation using improved covariance descriptors by regularization and adaptive feature extraction;
- clustering-based generative model updating method;
- online Partial Least Squares model learning methods that has constant time and space complexities;
- cascaded generative and discriminative object appearance models with a further extension that accommodates multiple models.

Fusing multiple features is important. We studied the region covariance descriptor from a machine learning perspective and examined its generalization ability. We reveal that small eigenvalues in the eigenspectrum of the conventional region covariance descriptor may cause large disturbance and thus degrade the generalization ability of the descriptor. We then proposed two improvements by regularization and PCA dimension reduction, resulting in two variants of the conventional region covariance descriptor called regularized covariance descriptor and adaptive covariance descriptor respectively. Effectiveness of these variants for improving generalization ability in case of small eigenvalues

is verified by empirical experiments. In addition, the adaptive covariance descriptor can achieve a significant time gain benefited from its reduced dimensionality.

For visual tracking, model updating is necessary in order to adapt the model to changes. Updating using reliable samples is crucial. We proposed to select a group of reliable samples collected in a short period to update the model. Sample selection is performed by feature space analysis using the mean-shift density gradient estimation procedure. Experiments showed that this selective updating strategy is able to keep the model up to date and to help preventing the model from being contaminated, particularly in case of short term occlusion or absence (out of the field-of-view).

We also exploited discriminative models and employed PLS as a tool for building discriminative appearance model. As tracking is an online process, we developed new online methods to incrementally update the PLS model. A more general weighted version is proposed as well. These methods have constant time and space complexities. We expect that the online PLS methods presented in this thesis could find more applications outside visual tracking. In fact, these online PLS methods can serve as time and space saving alternatives wherever PLS-1 could be applied.

Machine learning for visual tracking needs the model to ensure high prediction accuracy. Besides, it requires handling appearance changes which usually resorts to online learning. Online learning methods face the intrinsic problem of the tradeoff between stability and adaptivity. Since it is very hard for one appearance model to achieve all these merits, we have proposed to combine multiple models as a promising way to achieve this goal.

We proposed to cascade discriminative models after generative models to ensure detection accuracy. Generative models possess reconstructive ability while suffering from a lack of discriminative information between similar samples. Discriminative models on the other hand focus on discriminative information while discarding most reconstructive information. Cascading them in a such manner can make the detection structure both reconstructive and discriminative. High prediction accuracy can thus be expected. To further increase robustness and balance between stability and adaptivity, we further proposed an augmented version by integrating multiple generative models and multiple discriminative models into the cascade detection structure. Typically, tradeoff between stability and adaptivity can be achieved by employing proper thresholds and learning rates in different layers.

Our empirical results on self-captured sequences showed that a performance boost is achieved by integrating multiples models. Compared to state-of-the-art tracking methods on public sequences for long-term tracking, our method achieved superior average performance.

6.2 Limitations and perspectives

There are some open problems that our system did not focus on. Although full occlusions and out of the field-of-views can be well handled by our system, a gradual partial to full occlusion might degrade the performance of our system. We expect that integrating the merits of fragment/part-based representations as in [136, 137] may provide insights to overcome this problem. The non-negative matrix factorization [138] is also worth exploiting in this situation.

Another problem is the requirement for real-time processing, which is necessary for practical tracking system. Our current system employs the Intel Thread Building Block (TBB) to take advantage of parallel computing on modern multi-thread CPUs. We expect that a hybrid CPU/GPU parallel implementation could further boost up the speed of the tracking system and thus enable more advanced machine learning techniques to be applied.

In addition to part-based methods to circumvent partial occlusion, we present below some other perspectives that may be helpful for building robust visual tracking systems.

- Robust statistics can be naturally incorporated to increase the robustness of the tracking system. In particular, the Least Absolute Deviation (LAD) is generally recognized to be more robust than Least Squares (LS). It is plausible that when applied properly, the variants of popular machine learning techniques using ℓ_1 norm, e.g. ℓ_1-PCA, ℓ_1-LDA, or ℓ_1-PLS etc., may yield better tracking performances.

- Sparse representation for classification has gained great attentions in the recent years. In this thesis, we didn't exploit this family of methods. Although a number of works have been proposed in the literature [22, 67], further investigation in this direction may still be of interest.

- Machine learning for visual tracking faces the intrinsic problem of asymmetric (positive versus negative) training and testing samples. In this thesis, we tackle this problem by the cascaded classifier structure. It is also possible to explicitly build asymmetric classifiers. Some related works can be found in [112, 139].

- Recently, a new approach called self-paced learning that can retrospectively edit and select previous frames for learning is proposed in [140] and is shown to be effective for online adjusting the object appearance model. Further researches exploiting this idea might be promising.

Appendix A

Derivation of Equation (4.32)

See Fig. A.1.

For notation convenience, we denote $N(X_1)$ as N_1 and $N(X_2)$ as N_2.

$$\begin{aligned}
S_{xx} &= \sum_{i=1}^{N_1} f_1(x_i - \mu(X))(x_i - \mu(X))^\top + \sum_{i=N_1+1}^{N_1+N_2} f_2(x_i - \mu(X))(x_i - \mu(X))^\top \\
&= \sum_{i=1}^{N_1} f_1(x_i - \mu(X_1) + \mu(X_1) - \mu(X))(x_i - \mu(X_1) + \mu(X_1) - \mu(X))^\top \\
&\quad + \sum_{i=N_1+1}^{N_1+N_2} f_2(x_i - \mu(X_2) + \mu(X_2) - \mu(X))(x_i - \mu(X_2) + \mu(X_2) - \mu(X))^\top \\
&= f_1 \sum_{i=1}^{N_1}(x_i - \mu(X_1))(x_i - \mu(X_1))^\top + f_1 N_1(\mu(X_1) - \mu(X))(\mu(X_1) - \mu(X))^\top \\
&\quad + \sum_{i=N_1+1}^{N_1+N_2} f_2(x_i - \mu(X_2))(x_i - \mu(X_2))^\top + f_2 N_2(\mu(X_2) - \mu(X))(\mu(X_2) - \mu(X))^\top
\end{aligned}$$

By definition, we have

$$\begin{aligned}
S_{xx1} &= \sum_{i=1}^{N_1}(x_i - \mu(X_1))(x_i - \mu(X_1))^\top \\
S_{xx2} &= \sum_{i=N_1+1}^{N_1+N_2}(x_i - \mu(X_2))(x_i - \mu(X_2))^\top,
\end{aligned}$$

which yield

$$\begin{aligned}
S_{xx} &= f_1 S_{xx1} + f_1 N_1(\mu(X_1) - \mu(X))(\mu(X_1) - \mu(X))^\top + f_2 S_{xx2} + f_2 N_2(\mu(X_2) - \mu(X))(\mu(X_2) - \mu(X))^\top \\
&= f_1 S_{xx1} + f_2 S_{xx2} + f_1 N_1(\mu(X_1) - \mu(X))(\mu(X_1) - \mu(X))^\top + f_2 N_2(\mu(X_2) - \mu(X))(\mu(X_2) - \mu(X))^\top.
\end{aligned}$$

By further plugging

$$\mu(X) = \frac{f_1 N_1 \mu(X_1) + f_2 N_2 \mu(X_2)}{f_1 N_1 + f_2 N_2},$$

we have

$$\begin{aligned}
S_{xx} &= f_1 S_{xx1} + f_2 S_{xx2} + f_1 N_1(\mu(X_1) - \frac{f_1 N_1 \mu(X_1) + f_2 N_2 \mu(X_2)}{f_1 N_1 + f_2 N_2})(\mu(X_1) - \frac{f_1 N_1 \mu(X_1) + f_2 N_2 \mu(X_2)}{f_1 N_1 + f_2 N_2})^\top \\
&\quad + f_2 N_2(\mu(X_2) - \frac{f_1 N_1 \mu(X_1) + f_2 N_2 \mu(X_2)}{f_1 N_1 + f_2 N_2})(\mu(X_2) - \frac{f_1 N_1 \mu(X_1) + f_2 N_2 \mu(X_2)}{f_1 N_1 + f_2 N_2})^\top \\
&= f_1 S_{xx1} + f_2 S_{xx2} + \frac{f - 1 N_1 (f_2 N_2)^2}{(f_1 N_1 + f_2 N_2)^2}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top \\
&\quad + \frac{f^2 N_2 (f_1 N_1)^2}{(f_1 N_1 + f_2 N_2)^2}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top \\
&= f_1 S_{xx1} + f_2 S_{xx2} + \frac{f_1 N_1 f_2 N_2}{f_1 N_1 + f_2 N_2}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top \\
&= f_1 S_{xx1} + f_2 S_{xx2} + \frac{f_1 f_2 N(X_1) N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top.
\end{aligned}$$

FIGURE A.1: Derivation of Equation (4.32)

Appendix B

Résumé Etendu

B.1 Introduction

Le suivi visuel est une tâche fondamentale pour une variété d'applications de vision par ordinateur. Dans ce travail, nous nous concentrons sur le problème de la poursuite d'un objet quelconque dans une séquence vidéo sans aucune connaissance préalable autre que l'emplacement de l'objet dans la première image. Ce problème est difficile, car la cible peut subir diverses variations, causées par les occultations, les changements d'éclairage, etc. Il est encore plus difficile dans des environnements non contraints où la cible peut être totalement occlue ou quitte et puis réapparait dans le champ de vue de la caméra.

Dans le problème de suivi en mode "long terme", le mouvement n'est pas une information pertinente. La construction d'un modèle d'apparence fiable est une étape de toute première importance. Récemment, un certain nombre de méthodes ont abordé le problème de suivi par "suivi-par-détection". Un classifieur ou une variable explicative est appris en ligne pour s'adapter aux changements de la cible et aux variations du fond. Malgré leur succès, un certain nombre de problèmes est soulevé par ces méthodes. Le premier est la sélection d'échantillons dans l'image courante, la plupart des approches utilisent un échantillonnage aléatoire pour obtenir des échantillons négatifs. Le second problème est le compromis entre la stabilité et l'adaptabilité, comme les deux propriétés sont contradictoires entre elles. Une faiblesse de l'adaptabilité entraînerait l'incapacité du traceur à suivre les variations de l'objet et du fond tandis que le manque de stabilité au problème connu de "drifting". Le troisième problème est l'aspect asymétrique du classifieur ou de la variable explicative. Pour le suivi d'un seul objet, on se trouve en général dans le cas où des échantillons négatifs sont beaucoup plus nombreux que les échantillons positifs.

Inspiré par le détecteur de visage en cascade de [133], nous proposons de traiter ces problèmes en intégrant des modèles d'apparence multiples qui sont organisés en une structure de détection en cascade et mis à jour en ligne. La première contribution de ce travail consiste à proposer le cadre en cascade qui intègre des modèles d'apparence génératifs et discriminatoires. Plus précisément, le modèle génératif filtre candidats les plus faciles rapidement dans le stade précoce à l'aide des caractéristiques simples et conserve quelques candidats les plus prometteurs. Le modèle discriminatoire alors ré'evaluer ces échantillons à l'aide des descripteurs de région plus sophistiqués par les moindres carrés partiels (PLS) d'analyse discriminante. En organisant les modèles de la structure en cascade, l'évaluation de la confiance d'un échantillon de candidat peut être particulièrement efficace. Après un examen de travaux connexes dans la Section B.2, nous allons illustrer les détails des modèles d'apparence en cascade dans la Section B.4.

Notre motivation pour le mélange modèles génératifs et discriminatoires de la manière en cascade peut être expliqué en trois plis. Tout d'abord, la structure en cascade ne peut lutter efficacement le problème de l'asymétrie de la formation et des données de test. Suivi d'objets (et de détection d'objet) a le problème intrinsèque des échantillons asymétriques, à savoir les instances cibles limitées sont les échantillons positifs alors que tout le reste de "monde" étant échantillons négatifs. En ce sens, la structure en cascade autre que la structure parallèle prévoit un recours pour traiter le problème de l'asymétrie. En second lieu, la structure en cascade est plus efficace, c'est à dire a moins complexité de calcul, que les configurations parallèles. Les approches hybrides qui combinent plusieurs modèles en parallèle, par exemple en utilisant la co-formation [96], ont généralement plus de complexité de calcul parce que tous les échantillons de candidats doivent être évalués par chaque modèle avant de prendre une décision finale. En outre, ces approches sont encore confrontés au problème de déséquilibre des données, ce qui peut affecter les performances d'un classificateur discriminatoire. Troisièmement, l' hypothèse sous-jacente de l'apprentissage automatique est que les échantillons de formation et les échantillons d'essai sont issus de la même (même si inconnu) distribution. Dans la littérature, il a été une convention de sélectionner au hasard des échantillons pour la formation ou la mise à jour d'un modèle d'apparence discriminatoire. Nous soutenons que la formation et les tests sur des échantillons pré- sélectionnés par la couche de précédent dans une structure en cascade est plus susceptible de satisfaire l'hypothèse que la politique de sélection aléatoire. Par conséquent, une meilleure précision de détection peut être attendu dans une structure de détection en cascade. Les avantages de la structure de détection en cascade a été mis en évidence par le succès du système de détection de visage en cascade [133].

La deuxième contribution majeure est l'apprentissage en ligne des modèles d'apparence dans la structure de détection en cascade. Efficiente et efficace mise à jour des modèles

est l'élément clé pour les systèmes de suivi basées sur des modèles d'apparence. Dans la Section B.5, nous allons utiliser une méthode de mise à jour sur la base clustering pour le modèle génératif et de développer une méthode d'apprentissage de PLS en ligne pour le modèle discriminatoire. Les deux méthodes modèle de mise à jour peuvent être réalisées efficacement, ce qui rend notre système de suivi utile dans la pratique.

B.2 Etat de l'art

Génératives modèles représentent l' aspect d'un objet par l'apprentissage d'un modèle qui fournit une puissance suffisante reconstructive. Comme modèle des méthodes génératives de l'objet cible, les techniques employées par ces méthodes sont généralement sans surveillance, comme l'analyse en composantes principales (PCA), l'analyse en composantes indépendantes (ICA), l'attente maximisation (EM), etc suivi est alors exprimé en trouver le plus proche apparition objet au modèle. Quelques exemples d'algorithmes de suivi générateurs peuvent être trouvés dans [141], [56] et [21]. Pour s'adapter aux changements d'apparence, le modèle d'apparence est souvent mis à jour en ligne comme dans [21] et [56].

D'autre part, les méthodes discriminatoires cherchent à trouver une frontière de décision qu'il est préférable de séparer l'objet cible à partir de l'arrière-plan. Ceux-ci peuvent être appelés "suivi par la détection" méthodes comme proposé dans [1, 4, 19, 73, 77, 79, 80]. Techniques employées par les méthodes discriminatoires sont généralement l'analyse discriminante linéaire (LDA), machine à vecteurs de support (SVM), machine à vecteur de pertinence (RVM) [70], stimulant ainsi que leurs variantes. Lorsqu'il est correctement formé, les méthodes discriminatoires peuvent démontrer la robustesse pour éviter de distraction dans le fond, comme la différence de leurs homologues génératifs.

Visant à fusionner les avantages des deux méthodes, génératives et discriminatoires, plusieurs approches hybrides [88–90] ont été proposées. [88] décrit un modèle hybride où un sous-ensemble de grande dimension des paramètres sont formés pour maximiser la probabilité générative, et un autre petit sous-ensemble de paramètres sont discriminative formés pour maximiser la probabilité conditionnelle. Dans [89], Lin et al. forment un modèle en optimisant une combinaison convexe de la génératrice et les fonctions discriminatoires objectifs. [90] a proposé une Hybrides principes de Generative et modèles discriminatoires qui a montré que lorsque la fourniture de données de formation marqué est limité, la performance optimale correspond à un équilibre entre l'aspect purement générative et purement discriminatoire.

Récemment, une tendance à la combinaison de plusieurs trackers ou l'intégration de trackers avec des détecteurs a montré prometteur. Méthodes d'apprentissage en ligne semi-supervisés sont couramment utilisés dans ces méthodes. Dans [96], deux classificateurs avec des fonctions indépendantes sont co-formés dans ligne support vector machines. Les prévisions à partir des caractéristiques différentes sont fusionnées par combinaison de la carte de confiance de chaque classificateur à l'aide d'une méthode de pondération du classifieur, ce qui entraîne un classificateur final qui se comporte mieux que tout classificateur individuel. De même, Yu et al. dans [5] proposent à la co-formation en ligne d'un modèle génératif global et d'un modèle discriminatoire local. Pour activer la nouvelle acquisition, le modèle génératif utilise un certain nombre de sous-espaces de faibles dimensions linéaires pour coder toutes les variations d'apparence qui ont été vus. [12] proposent un filtre à particules en cascade avec des observateurs discriminatoires des durées de vie différentes pour suivre en vidéo à faible cadence. Bien que similaire à notre approche dans la structure, la formation en ligne est nécessaire pour cette méthode pour apprendre une longue durée de vie observateur. Dans [98], Kwon and Lee proposent de goûter à des modèles de mouvement et plusieurs modèles d'apparence multiples et de les intégrer à travers une chaîne interactive de Markov Monte-Carlo (IMCMC) cadre. Le modèle global d'observation est décomposé en de multiples modèles d'observation de base qui sont construites par clairsemée analyse en composantes principales (SPCA) d'un ensemble de modèles d'entités portant un aspect spécifique de l'objet. Le modèle de mouvement est également représentée par la combinaison de plusieurs modèles de mouvement de base, dont chacun couvre un type de mouvement différent.

Au lieu d'utiliser l'apprentissage semi-supervisé, [99] propose d'augmenter ligne (supervision) Méthode d'apprentissage avec des approches complémentaires de suivi pour obtenir des résultats plus stables. Trois trackers de différents degrés de l'adaptabilité sont combinés: un modèle de modèle simple comme un composant non adaptatif stable, un suivi moyen de décalage en fonction de flot optique comme élément hautement adaptative et une forêt aléatoire en ligne apparence que modérément adaptative de l'apprenant en fonction. De même, Kalal et al. proposent dans [6] un cadre de Tracking-Learning-Detection (TLD) où un tracker médian-Flow (pyramidale Lucas-Kanade traqueur [102] étendu avec vérification des erreurs avant-arrière) est combiné avec classificateur en ligne appris. Le point culminant dans le cadre de TLD est le composant penchant appelé apprentissage PN qui exploite à la fois la structure temporelle et spatiale dans une vidéo pour améliorer progressivement la précision du classificateur. Plus récemment, un cadre d'ensemble est proposé dans [103] pour le suivi multi-cible qui choisit de manière optimale résultat de poursuite de cible de celle de trackers indépendants et un détecteur à

chaque pas de temps. Sélection optimale est obtenue par une étape de liaison de données hiérarchique avec des paramètres discriminative formés à partir d'un cadre max-marge.

Nous renvoyons les lecteurs à une enquête de suivi visuel dans [14] et un examen des progrès récents et des tendances dans [15] pour plus d'informations.

B.3 Vue d'ensemble du système

B.3.1 Inférence bayésienne séquentielle pour le suivi visuel

Nous formulons le problème de suivi visuel comme un problème d'estimation d'état d'une manière similaire à celle de [23] et [21] incrementalpca. L'état variable $Theta_t$ décrit les paramètres de mouvement affine de la cible au temps t. Dans ce travail, nous considérons la variable d'état notée par quatre paramètres de transformation affines que $\Theta_t = (x_t, y_t, s_t, \varphi_t)$, où x_t, y_t, s_t, φ_t désignent x, y coordonnées, facteur d'échelle et l'aspect ratio respectivement.

Étant donné un ensemble d'images observées $O_t = \{O_1, \cdots, o_t\}$, nous cherchons à estimer la valeur de l'état caché variable $Theta_t$. En supposant une transition markovienne de l'Etat et en utilisant le théorème de Bayes, nous avons l'équation récursive suivante

$$p(\Theta_t|O_t) \propto p(o_t|\Theta_t) \int p(\Theta_t|\Theta_{t-1}) p(\Theta_{t-1}|O_{t-1}) d\Theta_{t-1} \tag{B.1}$$

où $p(\Theta_t|\Theta_{t-1})$ est le modèle de transition d'état, $p(o_t|\Theta_t)$ est le modèle d'observation et $p(\Theta_{t-1}|O_{t-1})$ est la probabilité a posteriori de mise à jour de manière récursive avec le temps. L'estimation de l'état $\hat{\Theta}_t$ est alors déterminé que le maximum de probabilité a priori (MAP) estimation, c'est à dire

$$\hat{\Theta}_t = \Theta_t^{map} = \arg\max_{\Theta_t} p(\Theta_t|O_t). \tag{B.2}$$

B.3.2 Vue d'ensemble du système

L'équation d'inférence (B.1) est régie par la transition modèle $p(\Theta_t|\Theta_{t-1})$ qui indique la corrélation temporelle des résultats de suivi de trames consécutives, et le modèle d'observation $p(o_t|\Theta_t)$ qui évalue la probabilité de Θ_t observant o_t.

Il a été souligné dans [140] que dans des circonstances difficiles de suivi, des modèles d'apparence fiables sont plus importantes que les modèles de mouvement complexes. Nous adoptons donc un modèle de transition simple mais efficace. Plus précisément, à chaque image, nous prélevons un certain nombre de particules selon une distribution

gaussienne qui indépendamment des modèles de chaque paramètre Θ_t autour de la cible état précédemment estimé $\hat{\Theta}_{t-1}$ comme

$$p(\Theta_t|\Theta_{t-1}) = N(\Theta_t; \hat{\Theta}_{t-1}, \Psi) \tag{B.3}$$

où Ψ est une matrice de covariance diagonale dont les éléments sont les variances correspondantes des paramètres affines respectives, c'est-à-dire $\sigma_x^2, \sigma_y^2, \sigma_s^2, \sigma_\varphi^2$. Pendant ce temps, le poids de chaque particule est également réinitialisé à chaque trame. Par conséquent, la probabilité postérieure $p(\Theta_t|O_t)$ devient entièrement dominée par le modèle d'observation $p(o_t|\Theta_t)$. L'estimation de l'état $\hat{\Theta}_t$ peut donc être obtenue en utilisant l'expression simplifiée

$$\hat{\Theta}_t = \arg\max_{\Theta_t} p(o_t|\Theta_t). \tag{B.4}$$

Nous concentrons sur le modèle d'observation dans ce qui suit car il est essentiel pour déterminer la correction de l'image qui est la plus susceptible d'être la cible d'intérêt. En particulier, nous proposons d'utiliser une cascade de modèles d'apparence génératives et discriminatoires comme notre modèle d'observation pour mesurer la probabilité. En résumé, le modèle d'observation proposé peut être considérée comme une cascade de deux modèles d'apparence (une génératif et l'autre discriminatoire) qui travaillent de manière séquentielle pour stimuler la performance du modèle combiné. L'intuition derrière le tracker en cascade est qu'un modèle génératif peut faire la confusion entre la cible et les régions de fond similaires de temps en temps, il est préférable de tenir compte de plusieurs candidats les plus en vue, aussi longtemps que la véritable cible est inclus dans cette collection. Le modèle discriminatoire apprend alors une frontière de décision délicate à distinguer la cible de ses rivaux. D'un autre point de vue, le modèle génératif fournit le modèle sélectif avec des échantillons de formation et d'essais soigneusement sélectionnés tandis que le modèle discriminatoire sert la ré-évaluation/raffinage des résultats du modèle génératif.

Lors de la conception des deux types de modèles d'apparence, certaines considérations sont prises. Pour des raisons de rapidité, nous utilisons certaines caractéristiques simples qui ne coûtent pas cher à calculer pour le modèle génératif car il évalue tous les échantillons possibles. En revanche, pour le modèle discriminatoire, nous employons des caractéristiques plus sophistiquées pour garantir l'exactitude, parce que les caractéristiques doivent contenir des informations riches qui peuvent distinguer les différences subtiles entre la vraie cible et les "distracteurs" semblables à lui.

Nous présentons une vue d'ensemble du cadre de poursuite comme cela est représenté sur la Fig. B.1. Les détails du modèle d'observation en cascade seront illustrés dans la

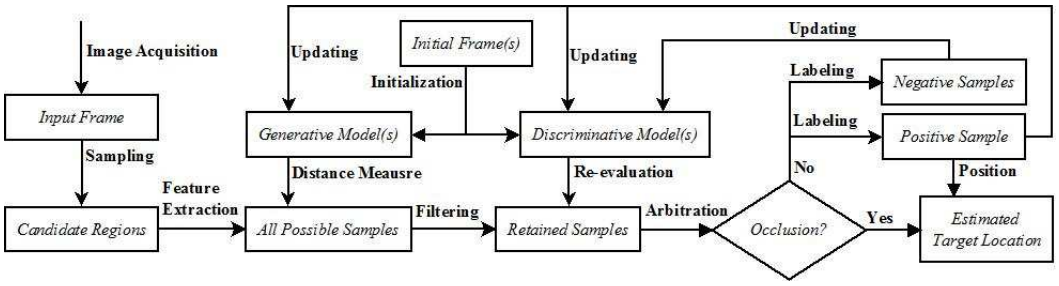


FIGURE B.1: Vue d'Ensemble du Système.

section suivante.

B.4 Suivi par des modèles d'apparence en cascade

B.4.1 Sélection des échantillons par le modèle d'apparence génératif

La probabilité de chaque échantillon proposé par le modèle dynamique est tout d'abord mesurée par le modèle génératif. Le modèle génératif est nécessaire pour être à la fois efficace et efficiente. Dans notre implémentation, nous considérons le vecteur de caractéristique obtenue par concaténation de 16-bin intensité des histogrammes d'une pyramide spatiale de 4 niveaux que la représentation de la région. Plus précisément, à chaque niveau L, le patch est divisé en $L \times L$ cellules résultant en un vecteur de caractéristique de 480 dimensions. Le modèle génératif M_g est donc une représentation de l'histogramme vecteur-forme. Le modèle initial M_g^0 est générée par le modèle cible annoté (ou détectée) dans la première image. Nous allons aborder l'évolution de M_g dans la Section B.5.1.

L'évaluation de la probabilité de l'échantillon est effectuée par un processus de correspondance de fonction qui calcule la distance entre la caractéristique de l'échantillon candidat et le modèle d'observation génératif. Dans ce travail, nous adoptons la distance du chi carré pour mesurer la différence entre un vecteur de fonction de test h_t et le modèle génératif M_g, qui est défini comme

$$d(H_t, M_g) = \sum_i \frac{(H_t(i) - M_g(i))^2}{0.5(H_t(i) + M_g(i))}. \tag{B.5}$$

Si seulement le modèle génératif a été utilisé, un seul meilleur match qui a la plus petite distance au modèle génératif serait choisi et être considérée comme le résultat de suivi pour la trame courante. Par contraste, notre système en cascade conserve quelques échantillons les plus prometteurs dans cette étape. La sélection de ces échantillons

importants est effectuée de manière itérative. Notons S_a comme l'ensemble de tous les échantillons possibles, nous choisissons la première fois en S_a l'échantillon s^* qui a la plus petite distance au modèle génératif et l'ajouter à S_R. Ensuite, nous retirons de S_a tous les échantillons qui ont région de chevauchement avec s^*. Les deux étapes sont répétées de manière itérative jusqu'à ce que le nombre voulu d'échantillons ont été recueillis dans S_R. On peut voir que les échantillons conservés en S_R sont mutuellement non-chevauchement. La raison pour la sélection non-chevauchement est de se promettant échantillons à l'échelle mondiale à partir de différents pics maximaux, éviter d'être pris au piège dans le quartier d'un seul maxima locaux. Nous résumons cette procédure de sélection dans l'algorithme 8, où $f(s)$ est le vecteur de caractéristiques (dans notre cas, l'histogramme de l'intensité enchaîné) extrait de l'échantillon s, $ROI(s)$ est sa boîte englobante.

Algorithm 8 Sélection des échantillons importants via le modèle génératif

Input: $\{S_a, N, S_r, M_g\}$
$S_a = \{$tous les échantillons possibles$\}$,
N: le nombre d'échantillons à conserver,
$S_r = \emptyset$: l'ensemble des échantillons retenus
Output: $\{S_r\}$
1: **while** $|S_r| \neq N$ **do**
2: $s^* \leftarrow \arg\min_{s \in S_a} d(f(s), M_g)$
3: $S_r \leftarrow S_r \cup \{s^*\}$
4: $S_a \leftarrow S_a$ - $\{s \in S_a | ROI(s) \cap ROI(s^*) \neq \emptyset\}$
5: **end while**

B.4.2 Ré-évaluation par le modèle discriminatoire

Les échantillons retenus à l'étape précédente sont réévalués par un modèle discriminatoire en utilisant les carrés partiels (PLS) moins [128] analyse discriminante adoptée dans ce travail. Nous allons examiner brièvement PLS dans B.4.2.1 et introduire une région descripteur plus sophistiqué pour la représentation de l'objet dans B.4.2.2. Formation du modèle d'aspect discriminatoire aide est adressée dans B.4.2.3 PLS. Enfin, la réévaluation des échantillons retenus par le modèle discriminatoire appris est présenté dans B.4.2.4.

B.4.2.1 Régression par moindres carrés partiels

PLS est un puissant outil statistique qui comprend la réduction de la dimension (d'extraction de caractéristiques compact) et des techniques de régression et considère la variable de réponse dans le processus.

Soit $X \in \Re^{N\times r}$ une matrice centrée moyen-de variables prédictives, avec des lignes correspondant à des observations et des colonnes à des variables et $Y \in \Re^{N\times m}$ la matrice de réponse moyenne centrée. PLS méthodes trouver de nouveaux espaces où la plupart des variations des échantillons observés peuvent être conservés, et les variables latentes acquises à partir de deux blocs sont plus corrélés à ceux dans les espaces originaux

$$\begin{aligned} X &= TP^\top + E \\ Y &= UQ^\top + F \end{aligned} \tag{B.6}$$

où $T \in \Re^{N\times p}$ and $U \in \Re^{N\times p}$ sont des facteurs (score, composants, variables latentes) matrices, $P \in \Re^{r\times p}$ and $Q \in \Re^{m\times p}$ sont en cours de chargement matrices, et $E \in \Re^{N\times r}$ et $F \in \Re^{N\times m}$ sont des termes d'erreur. Caractéristiques discriminatoires T sont extraites et la dimension est réduite lorsque $p < r$.

Pour la régression PLS, une relation linéaire entre le score vecteurs t et u existe, c'est à dire

$$U = TD + H \tag{B.7}$$

où $D \in \Re^{p\times p}$ est une matrice diagonale et H désigne la matrice des résidus. Il s'ensuit que Y est régressé par T comme

$$Y = TDQ^\top + (HQ^\top + F). \tag{B.8}$$

En injectant l'équation [127]

$$T = XW(P^\top W)^{-1} \tag{B.9}$$

où P est la matrice de chargement défini dans l'équation (B.6), l'équation de régression globale (à partir de X à Y) est écrit

$$Y = X(W(P^\top W)^{-1}DQ^\top) + F^*, \tag{B.10}$$

où $F^* = HQ^\top + F$ est le résiduel global. Le coefficient de régression globale β est donc formulée,

$$\beta = W(P^\top W)^{-1}DQ^\top. \tag{B.11}$$

Pour une fonction de test vecteur x_t, sa réponse de régression y_t est donc évaluée par

$$y_t = (x_t - \mu(X))^\top \beta + \mu(Y), \tag{B.12}$$

où $\mu(X)$ et $\mu(Y)$ sont les moyennes des échantillons de X et Y avant le moyen de centrage respectivement. D'autres détails concernant les méthodes de PLS peuvent être trouvés dans [128].

B.4.2.2 Descripteur de covariance adaptatif

Pour l'apprentissage du modèle discrimininatoire, nous utilisons une région descripteur plus sophistiqués pour assurer l'exactitude. Maintenant, c'est possible parce que le nombre d'échantillons conservés a été considérablement réduit et génère des descripteurs complexes pour ces moins d'échantillons devient abordable.

Le descripteur région de covariance proposée dans [8] fournit un moyen élégant de fusionner les caractéristiques spatiales et statistiques dans une matrice de covariance et a été largement utilisé dans diverses applications de vision par ordinateur [8, 60, 142].

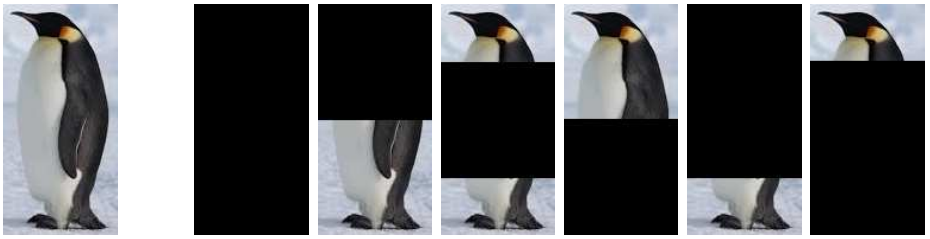
L'idée du descripteur de covariance est de représenter d'abord un pixel par un vecteur de caractéristiques pixelliques, comme par exemple les canaux de couleurs à partir d'une variété d'espaces de couleurs et des dérivés d'intensité de différents ordres. Dans un deuxième temps, on utilise une projection PCA pour extraire un certain nombre de caractéristiques pertinentes de cet ensemble de caractéristiques. La matrice de projection PCA est formée en utilisant l'ensemble de pixels de l'image r éférence de la cible dans l'image initiale en retenant les principaux vecteurs propres correspondant aux valeurs propres les plus importantes. Finalement, le descripteur de covariance adaptatif est généré par le calcul du descripteur de matrice de covariance en utilisant les caractéristiques pertinentes extraites.

En bref, le descripteur de covariance adaptatif effectue d'abord une extraction de caractéristiques par projection PCA et calcule le descripteur de covariance en utilisant les caractéristiques extraites alors. Il est adaptatif à un objet spécifique parce que la méthode d'extraction de caractéristiques, à savoir le vecteur moyen et la matrice de projection de l'PCA, est formé sur l'ensemble de pixel du référence de l'objet.

La projection PCA pendant le calcul du descripteur de covariance adaptatif préserve l'information dominante et supprime le bruit. Il est bien connu que les petites valeurs propres dans un eigenspectrum ne sont pas stables [110–112] et sont sensibles aux données de formation limitées. Retrait des dimensions non fiables peut alléger le problème de sur-apprentissage et donc d'améliorer la généralisation.

Un autre avantage du descripteur de covariance adaptatif est qu'elle génère une représentation plus compacte pour une région d'image. Cela peut faire les opérations suivantes sur la représentation, par exemple, formation ou la mise à jour du modèle discriminatoire et évaluer échantillon probabilité, beaucoup plus rapide. Bien que le descripteur de covariance adaptative nécessite calcul supplémentaire pour la formation de l'PCA et d'extraction de caractéristiques compacts en projetant les vecteurs de caractéristiques premières des principaux vecteurs propres, ces efforts de calcul supplémentaires sont

abordables et semble être bien utile. Tout d'abord, le calcul pour l'apprentissage de la PCA est négligeable, car elle est effectuée une seule fois à première image. Deuxièmement, le gain de vitesse dans les traitements ultérieur entraîné par la représentation compacte peut être considérablement supérieurs aux coûts de la projection de la PCA.



(a) (b) (c) (d) (e) (f) (g)

FIGURE B.2: Illustration d'un multi-taches représentation de région. Un objet (a) sur la gauche est représentée par six taches sur la droite, c'est à dire (b) l'ensemble région, (c) de la moitié supérieure, (d) de la moitié du milieu, c'est à dire de 1/4 à 3/4 de l'hauteur, (e) la moitié inférieure, (f)la 3/4 partie en haute (g)la 3/4 partie en bas.

Afin d'augmenter encore la robustesse, une région est représentée par six matrices de covariance d'adaptation de ses sous-régions 6 comme représenté sur la Fig. B.2. Le descripteur de covariance d'adaptation est en effet la matrice de covariance (bien que dans le sous-espace), qui est symétrique définie positive (SPD) et se trouve dans une variété Riemannienne. Utilization directement du descripteur pour l'analyse discriminante n'est pas approprié. En utilisant la transformation Log-Euclidien [61], nous poing transformer les descripteurs d'adaptation des 6 patchs $\log C_a^{(i)} (i = 1 \cdots 6)$ à l'espace euclidien comme $\log C_a^{(i)} (i = 1 \cdots 6)$, puis déplier chaque matrice et de les enchaîner à prendre une forme vectorielle. Nous notons que les matrices transformées $\log C_a$ sont toujours symétrique, seules matrices triangulaires supérieures sont utilisés. Par exemple, si 10 de 15 dimensions sont retenues pour le calcul du descripteur de covariance d'adaptation, la dimension de la représentation de vecteur final d'une région est de $10 \times (10+1)/2 \times 6 = 330$, tandis que le descripteur de covariance classique en utilisant 15 traits générerait un vecteur de taille $15 \times (15+1)/2 \times 6 = 720$.

B.4.2.3 Initialisation du modèle d'apparence

Pour notre application de suivi visuel, X dans la formulation PLS est la matrice formée par des représentations vectorielles accumulés calculé dans la section perméable qui décrivent les zones d'image de l'échantillon de formation. Un modèle discriminatoire peut être formé en mettant Y que les étiquettes binaires. Échantillons de formation sont de S_R qui sont collectées à chaque image lors de la poursuite avec les étiquettes correctement assignés. En particulier, nous maintenons un échantillon positif définir B_p

et un échantillon négatif définir B_n, qui sont initialisées comme des ensembles vides et sont finalement remplis d'échantillons marqués à partir de S_R lorsque le suivi évolue.

Dans la première image, nous utilisons l'échantillon positif annoté et un certain nombre d'échantillons négatifs pour initialiser le modèle discriminatoire. Les échantillons négatifs sont obtenus de la façon suivante. Tout d'abord, nous utilisons une méthode de recherche de glissement de fenêtre par force brute pour la recherche dans l'ensemble de l'image et de conserver quelques échantillons les plus prometteurs S_R. Comme le cible a été annoté de ce cadre, les autres échantillons dans S_R sont étiquetés comme des échantillons négatifs avec $y = -1$. Le modèle discriminatoire est alors initialisé par la formation PLS sur les ces échantillons. Pratiquement, nous utilisons un échantillons positifs et 20 échantillons négatifs pour initialiser le modèle discrimininatoire.

Pour la formation du modèle PLS, il peut être effectué soit par le NIPALS classique algorithme [124] ou par le plus rapide SIMPLS algorithme [125]. Comme alternative, nous allons introduire dans la Section B.5.2.1 une solution de PLS non-itératif qui peut permettre la mise à jour en ligne de modèle PLS. Après la formation, l'échantillon signifie $\mu(X)$, $\mu(Y)$ et le coefficient de régression β sont enregistrées pour évaluer plus tard échantillons d'essai.

Comme les progrès de temps, le modèle est mis à jour ponctuellement avec des échantillons nouvellement étiquetés. Ré-évaluation discriminatoire à l'aide du modèle PLS appris et l'étiquetage des nouveaux échantillons sont traités dans la suite et la mise à jour du modèle sera abordée dans la Section B.5.

B.4.2.4 Ré-évaluation par le modèle distriminatoire

Une fois le modèle discriminatoire est appris, il est utilisé à chaque trame de réévaluer la matrice de fonction X_r d'échantillons conservés dans S_R en utilisant (B.12) pour obtenir des scores de réponse Y_r. Il est à noter que ce processus de test est particulièrement rapide, car seulement un produit de point unique de vecteur de caractéristiques avec le coefficient de régression β est nécessaire (B.12) pour obtenir la réponse du PLS modèle de régression.

Selon le système d'étiquetage, la meilleure correspondance $\hat{s} \in S_R$ est choisi pour être celui associé à la plus grande valeur de la réponse $\hat{y}$. Un prédéfinie constante $\tau \in [0, 1)$ est utilisé comme seuil pour mesurer l'importance de $\hat{y}$. Si $\hat{y}$ a une valeur pas moins de τ, l'état de $\hat{s}$, noté $\Theta(\hat{s})$, est considéré comme l'estimation de $\hat{\Theta}_t$ et est donc sortie comme le résultat de suivi. $\hat{s}$ sera étiqueté comme un nouvel échantillon positif, c'est à dire avec $y = 1$ et seront ajoutés à la mémoire tampon de l'échantillon positif B_p, tandis que les autres échantillons dans S_R être étiquetés comme négatif (avec

$y = -1$) et être ajouté au tampon d'échantillon négatif B_n pour mettre à jour le modèle d'apparance ultérieurement. Sinon, c'est à dire si $\hat{y}$ est inférieur à τ, une **occlusion/absent** événement est déclaré, c'est l'objectif est considéré comme "invisible" dans ce cadre, et tous les échantillons dans S_R sont jetés.

Algorithme 9 résume la procédure de ré-évaluation par le modèle PLS.

Algorithm 9 Ré-évaluation par le modèle discriminitoire.

Input: $\{S_r, \beta, \tau, \hat{\Theta}_{t-1}, B_p, B_n, \mu(X), \mu(Y)\}$
Output: $\{\hat{\Theta}_t, B_p, B_n\}$
for $i = 1 \cdots |S_r|$ **do**
  $X_r^{(i)} \leftarrow$ caractéristique de la $i^{\text{ème}}$ échantillon $s^{(i)}$ dans S_r
  $Y_r^{(i)} \leftarrow (X_r^{(i)} - \mu(X))^\top \beta + \mu(Y)$
end for
$\hat{y} \leftarrow \max_{i=1...|S_r|} Y_r^{(i)}$
$\hat{s} \leftarrow \arg\max_{s \in S_r} Y_r(s)$
if $\hat{y} \geq \tau$ **then**
  $\hat{\Theta}_t \leftarrow \Theta(\hat{s})$
  $B_p \leftarrow B_p \cup \hat{s}$
  $B_n \leftarrow B_n \cup S_r \backslash \hat{s}$
else
  Déclarer **occlusion/absent**
end if

B.5 Mise à jour des modèles

Tant le générateur et les modèles discriminatoires doivent être mis à jour que le suivi développe pour s'adapter aux changements de la cible et le fond.

Sauf le cas **occlusion/absent**, les échantillons sélectionnés dans S_r sont étiquetés comme indiqué dans la section précédente et sont utilisés pour mettre à jour les modèles d'apparence.

B.5.1 Mise à jour du modèle génératif

Le modèle génératif concerne les échantillons positifs seulement. Une ligne de base signifie la mise à jour peut être servi pour la mise à jour. Afin d'augmenter la robustesse, nous adoptons le système de mise à jour sur la base clustering. Au lieu de mettre à jour à chaque trame, la mise à jour sur la base clustering fonctionne moins fréquemment, par exemple, toutes les 10 images. Il recueille les échantillons positifs marqués dans cette période, c'est à dire les 10 derniers échantillons dans B_p, et effectue un regroupement

des sauts entre eux. Les valeurs aberrantes sont filtrés en ne retenant que le groupe le plus proche de (dont la moyenne a la plus petite distance par rapport au modèle de générateur de courant). Le modèle $\hat{M}_g$ est alors déterminé comme combinaison linéaire du modèle initial M_g^0 (modèle calculé à partir de l'image initiale), le modèle actuel M_g et la moyenne des échantillons à la groupe retenue $\bar{m_S}$ défini comme

$$\begin{aligned} &\hat{M}_g = \alpha_1 \cdot M_g^0 + \alpha_2 \cdot M_g + \alpha_3 \cdot \bar{M}_s, \\ &s.t. \quad \Sigma_{i=1}^{3} \alpha_i = 1.0, \\ &\qquad 0 \leq \alpha_i \leq 1.0; \; i = 1, 2, 3. \end{aligned} \tag{B.13}$$

De cette manière, un échantillon est utilisé pour mettre à jour le modèle génératif s'il survit à la filtration de l'échantillon par le modèle de générateur, la ré-évaluation par le modèle discriminative et également la sélection de la grappe par le regroupement des sauts. Cela permet de protéger le modèle de la contamination et donc d'augmenter la robustesse.

B.5.2 Mise à jour du modèle discriminatoire

Pour la mise à jour en ligne du modèle discriminatoire PLS, les PLS algorithmes classiques tels que l'algorithme de NIPALS [124, 126] et la SIMPLS algorithme [125] ne sont pas adaptés car ils reposent sur l'ensemble de la formation X et Y pour calculer la sortie W et β. Pour chaque mise à jour, ils auraient besoin de recalculer le modèle PLS en utilisant les échantillons accumulés jamais vu. Ils sont donc des algorithmes de traitement par lots. Comme suivi évolue, le stockage et les exigences de calcul des algorithmes de traitement par lots ne cessent d'augmenter.

Pour contourner ce problème, nous adoptons une solution non-itératif alternatif [129] pour calculer le modèle PLS. Au lieu d'utiliser la première bloc de données X et Y, la méthode non-itérative prend les deux dispersion matrices S_{xx} et s_{xy} en entrée pour calculer le modèle PLS, qui sont définis comme

$$\begin{aligned} S_{xx} &= \sum_{i=1}^{N} (x_i - \mu(X))(x_i - \mu(X))^\top \\ S_{xy} &= \sum_{i=1}^{N} (x_i - \mu(X))(y_i - \mu(Y))^\top \end{aligned}$$

où N est le nombre d'échantillons dans X (et Y) et $\mu(X)$ et $\mu(Y)$ sont des moyens d'échantillonnage de X et Y respectivement. Lorsque X et Y ont été moyenne centrée

a priori, nous avons $S_{xx} = X^\top X$ et $S_{xy} = X^\top Y$ (organisation X et Y par lignes correspondant à des observations et des colonnes à des variables). Depuis ces deux matrices de dispersion sont constants dans la taille (indépendant de N) et peuvent être mis à jour progressivement avec de nouveaux échantillons, un algorithme PLS incrémentale devient possible.

Nous allons brièvement présenter la solution de PLS non-itératif dans la Section B.5.2.1 et de proposer à la Section B.5.2.2 un algorithme PLS incrémentale qui peut mettre à jour le modèle PLS dans la constante de temps et la complexité de l'espace. Le modèle discriminatoire dans notre cadre de suivi est ensuite mis à jour en ligne en utilisant l'algorithme PLS incrémentale nouvellement développé.

B.5.2.1 Une Solution non-itérative de la régression PLS-1

Selon la solution de PLS non-itératif [129], la relation entre la matrice de poids W_a et une matrice Krylov est reconnu. Le Krylov matrice $K_r \in \Re^{r \times rm}$ de la paire (S_{xx}, S_{xy}) est définie comme

$$K_r = [S_{xy} \quad S_{xx}S_{xy} \quad S_{xx}^2 S_{xy} \quad \cdots \quad S_{xx}^{r-1} S_{xy}]. \tag{B.14}$$

Une matrice Krylov réduite $K_p \in \Re^{r \times pm}$ est formée par les colonnes de premier p ($1 \leq p \leq r$) colonnes de K_r:

$$K_p = [S_{xy} \quad S_{xx}S_{xy} \quad S_{xx}^2 S_{xy} \quad \cdots \quad S_{xx}^{p-1} S_{xy}]. \tag{B.15}$$

La non-itératif solution à PLS [129] révèle que pour Y univariée, c'est à dire quand $m = 1$, la matrice classique de pondération orthonormé $W_p \in \Re^{r \times p}$ avec p variables latentes et la matrice Krylov K_p couvrent le même espace de colonne. De plus, W_p peut être calculé directement en effectuant la décomposition QR (et prendre la partie Q) ou la procédure de Gram-Schmidt modifié sur K_p.

En outre, le coefficient de régression β peut être calculé par une formule directe soit en utilisant K_p

$$\beta_{K_p} = K_p (K_p^\top S_{xx} K_p)^{-1} K_p^\top S_{xy} \tag{B.16}$$

ou en utilizant W_p

$$\beta_{W_p} = W_p (W_a^\top S_{xx} W_p)^{-1} W_p^\top S_{xy}. \tag{B.17}$$

Les deux expressions β_{K_p} et β_{W_p} donnent des résultats identiques car K_p et W_p durée de la même sous-espace [129]. Ils correspondent aux moindres carrés partiels (PLS) de

régression en utilisant des variables latentes p lorsque $p < r$ et de réduire les moindres carrés ordinaires (MCO) (en supposant que $K_r^\top K_r$ est inversible) lorsque $p = r$.

Dans la pratique, la matrice Krylov explicitement formulé K_p dans l'équation (B.15) peut être mal conditionné en raison d'arrondir des erreurs lors du calcul des puissances de S_{xx}, surtout quand p est large. Cela affecte négativement la précision de la résultante W_p et β. Dans ce travail, nous proposons d'utiliser la méthode de la Arnoldi [130] pour extraire la base orthonormée de K_p directement à partir de S_{xx} et S_{xy}. La procédure de pseudo-code de la méthode de la Arnoldi pour le calcul de W_p est décrite dans l'algorithme 10, où $\|\cdot\|_F$ est la norme de Frobenius. Le coefficient de régression β peut

Algorithm 10 La méthode de Arnoldi pour calculer la matrice de poids orthonormé W_p

Input: S_{xx} and S_{xy}: les matrices de dispersion
p: le nombre de composantes conservées
Output: la matrice de poids W_p

$w_1 \leftarrow S_{xy}/||S_{xy}||_F$
for $i = 2 \cdots p$ **do**
  $w_i \leftarrow S_{xx} w_{i-1}$
  for $j = 1 \cdots i-1$ **do**
    $h_{j,i-1} \leftarrow w_j^\top w_i$
    $w_i \leftarrow w_i - h_{j,i-1} w_j$
  end for
  $h_{i,i-1} \leftarrow ||w_i||_F$
  $w_i \leftarrow \frac{w_i}{h_{i,i-1}}$
end for
$W_p = [w_1\ w_2\ \cdots\ w_p]$

ainsi être obtenue à l'aide du résultat W_p selon l'équation (B.17).

B.5.2.2 Méthode incrémentale pour la mise à jour du modèle PLS

Comme la sortie W et β de l'algorithme PLS peuvent être calculées directement par S_{xx} et S_{xy}, nous pouvons spécifier un modèle PLS appris à partir d'un ensemble d'échantillons de formation X et Y comme

$$\Theta(X, Y, p) = (N(X), \mu(X), \mu(Y), S_{xx}, S_{xy}, W, \beta), \tag{B.18}$$

où p est la dimension de variables latentes, $N(X)$ est le nombre d'échantillons de formation dans X, $\mu(X)$ et $\mu(Y)$ sont des moyens. W et β peuvent être calculées par S_{xx}, S_{xy} et p par l'algorithme 10 et l'équation (B.17) respectivement. En fait, les cinq premiers éléments ont codé toutes les informations pour calculer ou mettre à jour un modèle PLS.

L'avantage de l'adoption de ce modèle, c'est qu'il a la complexité de l'espace constant parce que tous les éléments du modèle ont une taille constante.

Nous proposons maintenant une méthode PLS de la mise à jour incrémentale roman. Supposons que nous avons formé un modèle PLS formation mis X_1 et Y_1 de dimension p_1. Le modèle est ainsi notée $\Theta(X_1, Y_1, p_1)$. Lorsque de nouveaux échantillons, X_2 et Y_2, sont disponibles, l'algorithme de mise à jour incrémentale vise à mettre à jour le modèle PLS Θ avec X_2 et Y_2 sans avoir recours à l'ensemble des données initiales X_1 et Y_1.

Tout d'abord, les cinq premiers éléments de $\Theta(X_2, Y_2, p_2)$ est calculé comme $N(X_2)$, $\mu(X_2)$, $\mu(Y_2)$, S_{xx2}, S_{xy2} respectivement. Mise à jour incrémentale de $N(X)$, $\mu(X)$ et $\mu(Y)$ est simple:

$$N(X) = N(X_1) + N(X_2); \tag{B.19}$$

$$\mu(X) = \frac{N(X_1)}{N(X)}\mu(X_1) + \frac{N(X_2)}{N(X)}\mu(X_2), \tag{B.20}$$

$$\mu(Y) = \frac{N(X_1)}{N(X)}\mu(Y_1) + \frac{N(X_2)}{N(X)}\mu(Y_2). \tag{B.21}$$

La dispersion matrice S_{xx} peut être mis à jour en utilisant l'équation suivante:

$$S_{xx} = S_{xx1} + S_{xx2} + \frac{N(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top. \tag{B.22}$$

De même, S_{xy} peut être mis à jour

$$S_{xy} = S_{xy1} + S_{xy2} + \frac{N(X_1)N(X_2)}{N(X)}(\mu(X_2) - \mu(X_1))(\mu(Y_1) - \mu(Y_2))^\top. \tag{B.23}$$

La matrice de poids W peut donc être mis à jour en utilisant la nouvelle mise à jour S_{xx} et S_{xy} en fonction de l'algorithme 10. Enfin, le coefficient de régression β est mis à jour par l'équation (B.17). Nous notons que, bien que p peut être différent des deux p_1 et p_2, aucune information n'est perdue, puisque le nombre d'échantillons, les moyens et les matrices de dispersion ont intégré toutes les informations nécessaires pour mettre à jour le modèle.

Dans le suivi visuel, lorsque la cible subit des changements d'apparence, il est probable que des observations récentes seront plus indicatif de son apparence que les anciens plus. Par conséquent, il peut être souhaitable de se concentrer davantage sur les images récemment acquises et la baisse du poids de la contribution d'observations antérieures. Une façon de modérer l'équilibre entre les anciennes et les nouvelles observations est d'intégrer un facteur d'oubli dans la mise à jour incrémentale de PLS, comme cela se fait dans [143] et [21]. L'idée clé est la "nombre effectif" d'une observation. Par défaut, toutes les observations ont le même poids de 1. Si un échantillon est affecté d'un poids de

2, le résultat serait le même que si nous avions répété cet exemple deux fois en comptant le nombre de l'échantillon, le calcul des moyens et les matrices de dispersion. À l'autre extrême, un point associé à un poids de 0 serait le résultat comme si elle n'avait pas été inclus dans le calcul du tout.

Pour l'application de suivi, à chaque mise à jour, nous pouvons pondérer les observations connues par un facteur scalaire $f \in [0, 1]$, où $f = 1$ indique que pas d'oubli se produit. Le nombre effectif d'échantillons $N(X)$ et les moyennes de l'échantillon $\mu(X)$, $\mu(Y)$ sont mis à jour à l'aide du facteur d'oubli f comme

$$N(X) = fN(X_1) + N(X_2), \tag{B.24}$$

$$\mu(X) = \frac{fN(X_1)}{N(X)}\mu(X_1) + \frac{N(X_2)}{N(X)}\mu(X_2), \tag{B.25}$$

$$\mu(Y) = \frac{fN(X_1)}{N(X)}\mu(Y_1) + \frac{N(X_2)}{N(X)}\mu(Y_2). \tag{B.26}$$

La matrice de dispersion S_{xx} peut être mis à jour en utilisant l'équation suivante:

$$S_{xx} = fS_{xx1} + S_{xx2} + \frac{fN(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(X_1) - \mu(X_2))^\top. \tag{B.27}$$

De même, S_{xy} est mis à jour avec facteur d'oubli f comme

$$S_{xy} = fS_{xy1} + S_{xy2} + \frac{fN(X_1)N(X_2)}{N(X)}(\mu(X_1) - \mu(X_2))(\mu(Y_1) - \mu(Y_2))^\top. \tag{B.28}$$

Enfin, le modèle de régression W et β peuvent être mises à jour via l'algorithme 10 et l'équation (B.17) respectivement en utilisant le nouveau S_{xx} et S_{xy} qui sont récemment mis à jour.

Comme indiqué dans [21], un avantage d'intégrer le facteur d'oubli dans la mise à jour de moyenne, c'est que la moyenne peut encore changer en réponse à de nouvelles observations, même si le nombre réel d'observations tend vers l'infini. Plus précisément, en utilisant $N(X) = fN(X) + N(X_2)$, le nombre effectif d'observations atteindra l'équilibre à $N(X) = N(X_2)/(1 - f)$. Par exemple, quand $f = 0.95$ et $N(X_2) = 4$ de nouvelles observations sont ajoutées à chaque mise à jour, la taille effective de l'histoire de l'observation abordera $N(X) = 100$.

Avec l'algorithme PLS incrémentale développé ci-dessus, nous mettons à jour le modèle discriminatoire à l'aide des nouveaux échantillons marqués à chaque trame sauf le **occlusion/absent** événement se produit. Le nombre de variables latentes pour le modèle PLS est resté inchangé pendant la mise à jour.

Bibliography

[1] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 260–267, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.215. URL http://dx.doi.org/10.1109/CVPR.2006.215.

[2] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 234–247, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88681-5.

[3] S. Stalder, H. Grabner, and L. Van Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *International Conference on Computer Vison Workshops*, 2009.

[4] Babenko Boris, Yang Ming-Hsuan, and Belongie Serge. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, August 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.226. URL http://dx.doi.org/10.1109/TPAMI.2010.226.

[5] Qian Yu, Thang Ba Dinh, and Gérard Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, pages 678–691, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88685-3.

[6] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.239.

[7] Arnold W. M. Smeulders, Dung M. Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints):1, 2013.

ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.230.

[8] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *Proc. 9th European Conf. on Computer Vision*, volume 2, pages 589–600, 2006.

[9] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1623264.1623280.

[10] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[11] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1195991. URL http://dx.doi.org/10.1109/TPAMI.2003.1195991.

[12] Yuan Li, Haizhou Ai, Takayoshi Yamashita, Shihong Lao, and Masato Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1728–1740, 2008. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.73.

[13] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(6):810–815, 2004.

[14] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), December 2006. ISSN 0360-0300. doi: 10.1145/1177352.1177355. URL http://doi.acm.org/10.1145/1177352.1177355.

[15] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomput.*, 74(18):3823–3831, November 2011. ISSN 0925-2312. doi: 10.1016/j.neucom.2011.07.024. URL http://dx.doi.org/10.1016/j.neucom.2011.07.024.

[16] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1195991. URL http://dx.doi.org/10.1109/TPAMI.2003.1195991.

[17] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255, February 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000011205.11775.fd. URL http://dx.doi.org/10.1023/B:VISI.0000011205.11775.fd.

[18] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR*, pages 1–8, 2008.

[19] Hieu T. Nguyen and Arnold W. M. Smeulders. Robust tracking using foreground-background texture discrimination. *International Journal of Computer Vision*, 69 (3):277–293, 2006.

[20] Sam Hare, Amir Saffari, and Phil Torr. Struck: Structured output tracking with kernels. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011.

[21] D. A. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *IJCV*, 77:125–141, 2008.

[22] Xue Mei and Haibin Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2259–72, November 2011. ISSN 1939-3539. doi: 10.1109/TPAMI.2011.66. URL http://www.ncbi.nlm.nih.gov/pubmed/21422491.

[23] Michael Isard and Andrew Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[24] M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 50:174–188, 2002.

[25] Shaohua Zhou, Rama Chellappa, and Baback Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, 13:1434–1456, 2004.

[26] J. Kwon and F.C. Park. Visual tracking via particle filtering on the affine group. *The International Journal of Robotics Research*, 29(2-3):198–217, 2010. (Special issue on robot vision).

[27] Luka Cehovin, Matej Kristan, and Ales Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Trans. Pattern Anal. Mach.*

Intell., 35(4):941–953, 2013. URL http://dblp.uni-trier.de/db/journals/pami/pami35.html#CehovinKL13.

[28] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[29] F. Porikli. Integral histograms in cartesian spaces. In *CVPR*, 2005.

[30] M. Sizintsev, K. Derpanis, and A. Hogu. Histogram-based search: a comparative study. In *CVPR*, 2008.

[31] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[32] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014. URL http://dx.doi.org/10.1016/j.cviu.2007.09.014.

[33] Qiang Zhu, Shai Avidan, Mei chen Yeh, and Kwang ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *In CVPR06*, pages 1491–1498, 2006.

[34] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:1–8, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2008.4587630.

[35] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[36] J. Van De Weijer, Th. Gevers, and A.D. Bagdanov. Boosting color saliency in image feature detection. *IEEE TRANS. PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 28:150–156, 2005.

[37] Alaa E. Abdel-Hakim and Aly A. Farag. Csift: A sift descriptor with color invariant characteristics. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1978–1983, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.95. URL http://dx.doi.org/10.1109/CVPR.2006.95.

[38] Gertjan J. Burghouts and Jan-Mark Geusebroek. Performance evaluation of local colour invariants. *Comput. Vis. Image Underst.*, 113(1):48–62, January 2009. ISSN 1077-3142. doi: 10.1016/j.cviu.2008.07.003. URL http://dx.doi.org/10.1016/j.cviu.2008.07.003.

[39] Koen van de Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32 (9):1582–1596, September 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.154. URL http://dx.doi.org/10.1109/TPAMI.2009.154.

[40] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(8):837–842, August 1996. ISSN 0162-8828. doi: 10.1109/34.531803. URL http://dx.doi.org/10.1109/34.531803.

[41] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, July 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017623. URL http://dx.doi.org/10.1109/TPAMI.2002.1017623.

[42] Yadong Mu, Shuicheng Yan, Yi Liu, Thomas S. Huang, and Bingfeng Zhou. Discriminative local binary patterns for human detection in personal album. In *CVPR*. IEEE Computer Society, 2008. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2008.html#MuYLHZ08.

[43] Jie Chen, Shiguang Shan, Chu He, Guoying Zhao, Matti Pietikainen, Xilin Chen, and Wen Gao. Wld: A robust local image descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1705–1720, 2010. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2009.155.

[44] A. Alahi, P. Vandergheynst, M. Bierlaire, and M. Kunt. Cascade of descriptors to detect and track objects across any network of cameras. *Computer Vision and Image Understanding*, 114(6):624–640, June 2010.

[45] W. R. Schwartz and L. S. Davis. Learning Discriminative Appearance-Based Models Using Partial Least Squares. In *Brazilian Symposium on Computer Graphics and Image Processing*, 2009.

[46] Aniruddha Kembhavi, Behjat Siddiquie, Roland Miezianko, Scott McCloskey, and Larry S. Davis. Incremental multiple kernel learning for object recognition. In *ICCV*, pages 638–645, 2009.

[47] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, CIVR '07, pages 401–408, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-733-9. doi: 10.1145/1282280.1282340. URL http://doi.acm.org/10.1145/1282280.1282340.

[48] A C Berg and J Malik. Geometric blur for template matching. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conf. on*, volume 1, pages I—-607—-I—-614 vol.1, 2001. doi: 10.1109/CVPR.2001.990529.

[49] Peter V. Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *ICCV*, pages 221–228. IEEE, 2009. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2009.html#GehlerN09.

[50] Michael E. Tipping and Chris M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.

[51] A. Hyvärinen and E. Oja. Independent component analysis: Algorithms and applications. *Neural Netw.*, 13(4-5):411–430, May 2000. ISSN 0893-6080. doi: 10.1016/S0893-6080(00)00026-5. URL http://dx.doi.org/10.1016/S0893-6080(00)00026-5.

[52] Geoffrey Mclachlan and David Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley-Interscience, 1 edition, October 2000. ISBN 9780471006268. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471006262.

[53] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, November 1996. ISSN 10535888. doi: 10.1109/79.543975. URL http://dx.doi.org/10.1109/79.543975.

[54] Richard Baraniuk. Compressive sensing. *IEEE Signal Processing Mag*, pages 118–120, 2007.

[55] Michael J. Black, David J. Fleet, and Yaser Yacoob. A framework for modeling appearance change in image sequences. In *ICCV*, 1998.

[56] Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. In *IEEE Conference on Computer Vision and and Pattern Recognition*, volume I, pages 415–422, 2001.

[57] Yongmin Li. On incremental and robust subspace learning. *Pattern Recognition*, 37:1509–1518, 2004.

[58] Danijel Skocaj and Ales Leonardis. Weighted and robust incremental method for subspace learning. In *ICCV*, pages 1494–1501, 2003.

[59] Kuang chih Lee and David Kriegman. Online learning of probabilistic appearance manifolds for video-based recognition and tracking. In *In Proc. of CVPR*, pages 852–859, 2005.

[60] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance tracking using model update based on lie algebra. volume 1, pages 728–735, Los Alamitos, CA, USA, 2006. IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2006.94.

[61] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on sysmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 29 1:328–347, 2006.

[62] Xi Li, Weiming Hu, Zhongfei Zhang, Xiaoqin Zhang, Mingliang Zhu, Jian Cheng, and Guan Luo. Visual tracking via incremental log-euclidean riemannian subspace learning. In *In Proceedings IEEE Conference Computer Vision and Pattern Recognition*, 2008.

[63] Xi Li, Weiming Hu, Zhongfei Zhang, Xiaoqin Zhang, and Guan Luo. Robust visual tracking based on incremental tensor subspace learning. In *ICCV*, 2007.

[64] Yi Wu, Jian Cheng, Jinqiao Wang, and Hanqing Lu. Real-time visual tracking via incremental covariance tensor learning. In *ICCV*, pages 1631–1638. IEEE, 2009. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2009.html#WuCWL09.

[65] Xue Mei and Haibin Ling. Robust visual tracking using ℓ1 minimization. In *ICCV*, pages 1436–1443, 2009.

[66] Hanxi Li, Chunhua Shen, and Qinfeng Shi. Real-time visual tracking using compressive sensing. In *CVPR*, pages 1305–1312. IEEE, 2011. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2011.html#LiSS11.

[67] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 864–877, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33711-6. doi: 10.1007/978-3-642-33712-3_62. URL http://dx.doi.org/10.1007/978-3-642-33712-3_62.

[68] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[69] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20 (3):273–297, September 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411.

[70] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, September 2001. ISSN 1532-4435. doi: 10.1162/15324430152748236. URL http://dx.doi.org/10.1162/15324430152748236.

[71] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[72] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324.

[73] Robert Collins, Yanxi Liu, and Marius Leordeanu. On-line selection of discriminative tracking features. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, October 2005.

[74] Jianyu Wang, Xilin Chen, and Wen Gao. Online selecting discriminative tracking features using particle filter. In *In Proc. CVPR*, pages 1037–1042, 2005.

[75] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference*, pages 6.1–6.10. BMVA Press, 2006. ISBN 1-901725-32-4. doi:10.5244/C.20.6.

[76] Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *3rd IEEE - ICCV Workshop on On-line Learning for Computer Vision*, 2009. URL http://www.ymer.org/papers/files/2009-OnlineRandomForests.pdf.

[77] Shai Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[78] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1623264.1623280.

[79] Oliver Williams, Andrew Blake, and Roberto Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and*

Machine Intelligence, 27(8):1292–1304, 2005. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.167.

[80] Shai Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, feb 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.35. URL http://dx.doi.org/10.1109/TPAMI.2007.35.

[81] Min Tian, Weiwei Zhang, and Fuqiang Liu. On-line ensemble svm for robust object tracking. In Yasushi Yagi, Sing Bing Kang, In-So Kweon, and Hongbin Zha, editors, *ACCV (1)*, volume 4843 of *Lecture Notes in Computer Science*, pages 355–364. Springer, 2007. ISBN 978-3-540-76385-7. URL http://dblp.uni-trier.de/db/conf/accv/accv2007-1.html#TianZL07.

[82] Xiaoqin Zhang, Weiming Hu, Stephen J. Maybank, and Xi Li. Graph based discriminative learning for robust and efficient object tracking. In *ICCV*, pages 1–8. IEEE, 2007. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2007.html#ZhangHML07.

[83] Meng Wang, Xian-Sheng Hua, Jinhui Tang, and Richang Hong. Beyond distance measurement: Constructing neighborhood similarity for video annotation. *IEEE Transactions on Multimedia*, 11(3):465–476, 2009. URL http://dblp.uni-trier.de/db/journals/tmm/tmm11.html#WangHTH09.

[84] Meng Wang, Xian-Sheng Hua, Richang Hong, Jinhui Tang, Guo-Jun Qi, and Yan Song. Unified video annotation via multigraph learning. *IEEE Trans. Circuits Syst. Video Techn.*, 19(5):733–746, 2009. URL http://dblp.uni-trier.de/db/journals/tcsv/tcsv19.html#WangHHTQS09.

[85] Ming Yang, Junsong Yuan, and Ying Wu. Spatial selection for attentional visual tracking. In *CVPR*, 2007.

[86] Julia A. Lasserre. Principled hybrids of generative and discriminative models. In *In CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 87–94. IEEE Computer Society, 2006.

[87] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 841–848. MIT Press, 2001. URL http://dblp.uni-trier.de/db/conf/nips/nips2001.html#NgJ01.

[88] Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew Mccallum. Classification with hybrid generative/discriminative models. In *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[89] Ruei-Sung Lin, David A. Ross, Jongwoo Lim, and Ming-Hsuan Yang. Adaptive discriminative generative model and its applications. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, 2004.

[90] Julia A. Lasserre. Principled hybrids of generative and discriminative models. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 87–94. IEEE Computer Society, 2006.

[91] Helmut Grabner, Peter M. Roth, and Horst Bischof. Eigenboosting: Combining discriminative and generative information. In *CVPR*. IEEE Computer Society, 2007. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2007.html#GrabnerRB07.

[92] T. E. Woodley, B. Stenger, and R. Cipolla. Tracking using online feature selection and a local generative model. In *Proceedings of the British Machine Vision Conference*, pages 86.1–86.10. BMVA Press, 2007. ISBN 1-901725-34-0. doi:10.5244/C.21.86.

[93] Chad Aeschliman, Johnny Park, and Avinash C. Kak. A probabilistic framework for joint segmentation and tracking. In *CVPR*, pages 1371–1378. IEEE, 2010. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#AeschlimanPK10.

[94] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

[95] Bernhard Zeisl, Christian Leistner, Amir Saffari, and Horst Bischof. On-line semi-supervised multiple-instance boosting. In *CVPR*, page 1879. IEEE, 2010. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#ZeislLSB10.

[96] Feng Tang, Shane Brennan, Qi Zhao, and Hai Tao. Co-tracking using semi-supervised support vector machines. In *Computer Vision, IEEE International Conference on*, volume 0, pages 1–8, Los Alamitos, CA, USA, 2007. IEEE Computer Society. ISBN 978-1-4244-1630-1. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2007.4408954.

[97] Michael D. Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *IEEE International Conference on Computer Vision*, October 2009.

[98] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[99] Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock, and Horst Bischof. PROST Parallel Robust Online Simple Tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010.

[100] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Proceedings of SPIE: Optical Pattern Recognition XII*, volume 4387, pages 95–102, March 2001. doi: 10.1117/12.421129. URL http://dx.doi.org/10.1117/12.421129.

[101] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic huber-l1 optical flow. In *BMVC*. British Machine Vision Association, 2009. URL http://dblp.uni-trier.de/db/conf/bmvc/bmvc2009.html#WerlbergerTPWCB09.

[102] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000.

[103] Xu Yan, Xuqing Wu, Ioannis A. Kakadiaris, and Shishir K. Shah. To track or to detect? an ensemble framework for optimal selection. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *ECCV*, volume 7576 of *Lecture Notes in Computer Science*, pages 594–607. Springer, 2012. ISBN 978-3-642-33714-7.

[104] W. Förstner and B. Moonen. A metric for covariance matrices. Technical report, Dept. of Geodesy and Geoinformatics, Stuttgart University, 1999.

[105] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 66: 41–66, 2006.

[106] X. Li, W. Hu, Z. Zhang, X. Zhang, and J. Cheng. Visual tracking via incremental log-euclidean riemannian subspace learning. In *CVPR*, 2008.

[107] A. Alahi, D. Marimon, M. Bierlaire, and M. Kunt. Object detection and matching with mobile cameras collaboratin with fixed cameras. In *Proc. 10th European Conference on Computer Vision*, pages 1523–1550, 2008.

[108] P. Cortez-Cargill, C. Undurraga-Rius, D. Mery-Quiroz, and A. Soto. Performance evaluation of the covariance descriptor for target detection. In *International Conference of the Chilean Computer Science Society*, pages 133–141, 2009.

[109] M. J. Metternich, M. Worring, and A. W. M. Smeulders. Color based tracing in real-life surveillance data. *Transactions on Data Hiding and Multimedia Security*, V, 2010. URL http://www.science.uva.nl/research/publications/2010/MetternichTDHMS2010a.

[110] Jerome H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989. ISSN 0162-1459 (print), 1537-274X (electronic).

[111] Xudong Jiang, Bappaditya Mandal, and Alex Kot. Eigenfeature regularization and extraction in face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):383–394, 2008. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2007.70708.

[112] Xudong Jiang. Asymmetric principal component and discriminant analyses for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):931–937,, 2009. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.258.

[113] C.R. Rao and H. Toutenburg. *Linear Models: Least Squares and Alternatives*. Springer series in statistics. Springer, 1999. ISBN 9780387988481. URL http://books.google.fr/books?id=mfsnCM_q_T8C.

[114] R. Collins, X. Zhou, and S. Teh. An open source tracking testbed and evluation website. In *PETS*, 2005.

[115] Chis Ding and Xiaofeng He. K-means clustering via principal component analysis. In *International Conference on Machine Learning*, Banff, Canada, 2004.

[116] Hongyuan Zha, Xiaofeng He, Chis Ding, and Horst Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1064, 2002.

[117] Yizong Cheng. Mean shift, mode seeking, and clustering. *TPAMI*, 17(8)(8):790–799, 1995.

[118] Nadav Ben-Haim, Boris Babenko, and Serge Belongie. Improving web-based image search via content based clustering. In *SLAM*, New York City, 2006.

[119] Yunpeng Liu, Guangwei Li, and Zelin Shi. Covariance tracking via geometric particle filtering. *EURASIP Journal on Advances in Signal Processing*, 2010:22:1–22:9, February 2010.

[120] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *IEEE International Conference on Computer Vision*, pages 24–31, 2009.

[121] W. R. Schwartz, H. Guo, J. Choi, and L. S. Davis. Face identification using large feature sets. *IEEE Transactions on Image Processing*, 21(4):2245–2255, 2012.

[122] G. Chiachia, N. Pinto, W. R. Schwartz, A. Rocha, A. X. Falcao, and D. Cox. Person-specific subspace analysis for unconstrained familiar face identification. In *British Machine Vision Conference*, 2012.

[123] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang. Object tracking via partial least squares analysis. *IEEE Transactions on Image Processing*, 21(10): 4454–4465, 2012.

[124] Herman Wold. Path models with latent variables: The nipals approach. In H M Blalock, A Aganbegian, F M Borodkin, R Boudon, and V Capecchi, editors, *Quantitative Sociology: International perspectives on mathematical and statistical modeling*, pages 307–357. Academic Press, 1975.

[125] Sijmen de Jong. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18:251–263, March 1993.

[126] Hervé Abdi. Partial least squares (PLS) regression. In *Encyclopedia for research methods for the social sciences*, pages 792–795. Sage, 2003.

[127] Rolf Manne. Analysis of two partial-least-squares algorithms for multivariate calibration. *Chemometrics and Intelligent Laboratory Systems*, 2(1):187–197, 1987.

[128] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *Lecture Notes in Computer Science*, volume 3940, pages 34–51, 2006.

[129] David di Ruscio. A weighted view on the partial least-squares algorithm. *Automatica*, 36:831–850, 2000.

[130] Walter Edwin Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(17):17–29, 1951.

[131] K. Bache and M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[132] Doug Gray, Shane Brennan, and Hai Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *In IEEE International Workshop on Performance Evaluation for Tracking and Surveillance, Rio de Janeiro*, 2007.

[133] Paul Viola and Michael J. Jones. Robust real-time face detection. *Internaltional Journal of Computer Vision*, 57(2):137–154, May 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000013087.49260.fb. URL http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb.

[134] Fatih Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 829–836, 2005.

[135] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *ICPR*, pages 2756–2759. IEEE, 2010.

[136] Luka Cehovin, Matej Kristan, and Ales Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):941–953, 2013. URL http://dblp.uni-trier.de/db/journals/pami/pami35.html#CehovinKL13.

[137] Rui Yao, Qinfeng Shi, Chunhua Shen, Yanning Zhang, and Anton van den Hengel. Part-based visual tracking with online latent structural learning. In *CVPR*, pages 2363–2370. IEEE, 2013. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2013.html#YaoSSZH13.

[138] Inderjit S. Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with bregman divergences. In *In: Neural Information Proc. Systems*, pages 283–290, 2005.

[139] Hai-Ni Qu, Guo-Zheng Li, and Wei-Sheng Xu. An asymmetric classifier based on partial least squares. *Pattern Recognition*, 43(10):3448 – 3457, 2010. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2010.05.002. URL http://www.sciencedirect.com/science/article/pii/S0031320310002037.

[140] James Steven Supančič III and Deva Ramanan. Self-Paced Learning for Long-Term Tracking. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2379–2386, June 2013. doi: 10.1109/CVPR.2013.308. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6619152.

[141] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, January 1998. ISSN 0920-5691. doi: 10.1023/A:1007939232436. URL http://dx.doi.org/10.1023/A:1007939232436.

[142] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2007.

[143] Avraham Levy and Michael Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8): 1371–1374, 2000.

Lei QIN

Doctorat : Optimisation et Sûreté des Systèmes

Année 2014

Algorithmes d'apprentissage en ligne pour le suivi visuel

Nous étudions le problème de suivi de cible dans une séquence vidéo sans aucune connaissance préalable autre qu'une référence annotée dans la première image. Pour résoudre ce problème, nous proposons une nouvelle méthode de suivi temps-réel se basant sur à la fois une représentation originale de l'objet à suivre (descripteur) et sur un algorithme adaptatif capable de suivre la cible même dans les conditions les plus difficiles comme le cas où la cible disparaît et réapparait dans le scène (ré-identification). Tout d'abord, pour la représentation d'une région de l'image à suivre dans le temps, nous proposons des améliorations au descripteur de covariance. Ce nouveau descripteur est capable d'extraire des caractéristiques spécifiques à la cible, tout en ayant la capacité à s'adapter aux variations de l'apparence de la cible. Ensuite, l'étape algorithmique consiste à mettre en cascade des modèles génératifs et des modèles discriminatoires afin d'exploiter conjointement leurs capacités à distinguer la cible des autres objets présents dans la scène. Les modèles génératifs sont déployés dans les premières couches afin d'éliminer les candidats les plus faciles alors que les modèles discriminatoires sont déployés dans les couches suivantes afin de distinguer la cibles des autres objets qui lui sont très similaires. L'analyse discriminante des moindres carrés partiels (AD-MCP) est employée pour la construction des modèles discriminatoires. Enfin, un nouvel algorithme d'apprentissage en ligne AD-MCP a été proposé pour la mise à jour incrémentale des modèles discriminatoires.

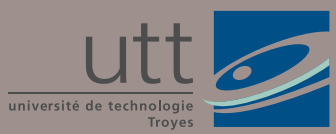
Mots clés : analyse multivariée - détection du signal - analyse de covariance – apprentissage automatique.

Online Machine Learning Methods for Visual Tracking

We study the challenging problem of tracking an arbitrary object in video sequences with no prior knowledge other than a template annotated in the first frame. To tackle this problem, we build a robust tracking system consisting of the following components. First, for image region representation, we propose some improvements to the region covariance descriptor. Characteristics of a specific object are taken into consideration, before constructing the covariance descriptor. Second, for building the object appearance model, we propose to combine the merits of both generative models and discriminative models by organizing them in a detection cascade. Specifically, generative models are deployed in the early layers for eliminating most easy candidates whereas discriminative models are in the later layers for distinguishing the object from a few similar "distracters". The Partial Least Squares Discriminant Analysis (PLS-DA) is employed for building the discriminative object appearance models. Third, for updating the generative models, we propose a weakly-supervised model updating method, which is based on cluster analysis using the mean-shift gradient density estimation procedure. Fourth, a novel online PLS-DA learning algorithm is developed for incrementally updating the discriminative models. The final tracking system that integrates all these building blocks exhibits good robustness for most challenges in visual tracking. Comparing results conducted in challenging video sequences showed that the proposed tracking system performs favorably with respect to a number of state-of-the-art methods.

Keywords: multivariate analysis - signal detection - analysis of covariance – machine learning.

Thèse réalisée en partenariat entre :



Ecole Doctorale "Sciences et Technologies"