People Re-identification in a Camera Network

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Ghadeer Shaaya

Supervised by

Dr. Tony Han

December 2012

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

People Re-identification in a Camera Network

presented by Ghadeer Shaaya,

a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Tony Han

Dr. Zhihai He

Dr. Duan Ye

To my family, thank you!

Especially mom and dad, a gigantic thinks.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks and gratitude to Dr. Tony Han, my academic advisor for his valuable guidance, support, encouragement and for all the facilities he provided to me.

I place on record, a special thanks to my family and true friends. You are great and I cannot be more appreciative to what you have done to me.

I take this opportunity to thank all the faculty members of the Department of Electrical and Computer Engineering for their help and support. I also thank all Dr. Han's lab members for their unlimited support and help.

I also place on record, my sense of gratitude to one and all who, directly and indirectly, have lent their helping hand in this venture.

Table of Contents

LIST OF FIGURES

ABSTRACT

In this research, we present an appearance based method for people re-identification. It consists in the extraction of two types of features related to human appearance, color histograms and SIFT features. Images are captured from surveillance videos. For every image, the two types of features are combined to create a two dimensional signature that represents the contained individual. The goal is to make this signature as distinctive as possible. The signatures are arranged into pairs to form positive examples (two images of the same individual) and negative examples (two images of two different individuals). Pairs are fed to a machine learning algorithm. The algorithm is trained to find the most discriminative model. AdaBoost is what we used to perform this task. The algorithm presented in this thesis has been tested on several datasets (ViPER, CAVIAR, 3DPes).

Chapter 1

Introduction

As a result of the emergent need for an intelligent system capable of analyzing human behavior, computer vision researchers have focused their efforts toward applications that offer automated video surveillance. Their particular focus is on people tracking, detection and recognition. Since such systems are used to guarantee the safety of people and places of interest (i.e. airports, metro stations, governmental institutes), developing security systems for detection and intervention is becoming an important necessity against the threats of terrorism or vandalism of property.

The growth of the computational and communication capabilities is setting new standards and breaking ground for surveillance applications and approaches,

making it easier to compute larger amounts of data faster. Prior to these advancements, cameras were set up in large scale networks, communicating with a main server through a wireless network. Retrieving this footage became a trivial task. Because of this, there became a greater need to use cameras to cover a wider area. The dispersed system can offer many advantages, such as scalability and fault tolerance. It can also present challenges due to the difficulty to integrate the information that are collected from multiple sources [4]. In multiple camera networks the challenging problem is the lack of temporal constrains when matching across non-overlapping fields of view in a camera network [1]. This problem is referred to in the literature of "Re-identification" [1, 2].

Many issues need to be taken care of while designing a practical video surveillance system. The most critical issue is handling the large amount of data streaming from all cameras at once. The system must be reliable at all the times. This is a very challenging task when searching for a person in many hours of footage.

Multiple cameras can easily flood the system with information. For a better understanding of the precise activity in the monitored area, a technique needs to

be developed to combine the data. This will help the analyst answer queries and retrieve particular data reliably in reasonable time.

An effective algorithm should be able to capture particular event characteristics in the monitored area. For example, it would be able to detect and describe an object moving across the camera field of view. These descriptors could then be used to establish connections among the objects detected in different cameras [3].

Usually, it is assumed that individuals wear the same clothes between the views of different cameras. The designed re-identification algorithm should consider number of factors, individual pose, viewpoint, illumination [4].

For an ideal surveillance system, the operator should be able to track a person whenever needed. We want to design a system that is capable of following one person's movement in a camera network; we want to be able to detect them when entering the field, to know where they are going or where they came from and to highlight their position. This problem has been discussed and solutions has been proposed and described as in [5, 6, 7, and 8].

The re-identification problem is different from the initial identification. In the re-identification system, there is no need to identify a person from days before. We only need to find a person within a small set of people that have been in the environment for the past few hours. Under this hypothesis we can assume that people do not change their appearance or clothes while crossing the field of surveillance.

Motivations and Goals

The growing concern for human safety is what keeps researchers dedicated to this field of study. When the camera is fixed and number of targets is minimal, simple algorithms can efficiently handle the problem of re-identifying people based on direction of movement and target location. As the amount of targets increase, it becomes harder to use a simple algorithm because it degrades the performance. This problem can be alleviated by the use of multiple hypotheses [9] with its efficient implementation [10]. However, as the size of the scene grows, the number of cameras needed to cover the area grows respectively to provide adequate coverage. This is what creates the problem of people re-identification,

the focus of this thesis. The ultimate goal of the system is to re-acquire targets, in other words, searching the camera network for the person of interest [2].

Lack of temporal information is one important challenge that leads to the use of an appearance model [2]. One choice of appearance model is a template, but it should be viewpoint and pose specific. If the angle of view point is fixed and known, fixable matching and alignment can be used [11, 12]. However this will not work with the human body considering it is a non-rigid object. If the problem is limited to the frontal viewpoint then one could use a triangle graph model [13]. While the template methods model the spatial layout of the object, histogram methods model its statistical properties [2]. Histogram is proven to be useful for a variety of tasks including tracking [14], texture classification [15] and pedestrian detection [16].

Much of the research has combined the template and the histogram methods for more reliable results. Past approaches include recording correlations in Correlograms [17], spatial positions in Spatiograms [18] or scale in multi-resolution histograms [19]. Both template and histogram methods suffer from problems with illumination changes, however, it has been shown that this can be handled by learning the brightness transfer function between cameras [20].

The appearance model that is presented in this thesis is a hybrid of texture and color histogram; however instead of designing the model ourselves, we let the machine do the work through a machine learning algorithm. This algorithm is capable of providing a maximum separation between the positive and the negative examples for a set of training data. The learned model is an ensemble of local features. The feature itself consists of feature channel, location and binning information and a likelihood ratio test for comparing corresponding features. Once we have the learned model, it provides the similarity function for comparing pairs of images. This function can be used for people re-identification.

Out of all the data available, training a model for people re-identification needs multiple images for the same individual. We have chosen to use three datasets (VIPer, CAVIAR and 3DPes) for this reason. VIPeR [21] contains two views of 632 pedestrians. CAVIAR [22] contains 1220 images for 72 individuals. 3DPes [23] contains 1012 snapshots of 200 people. Some examples from each dataset can be found in Figure (1.1).

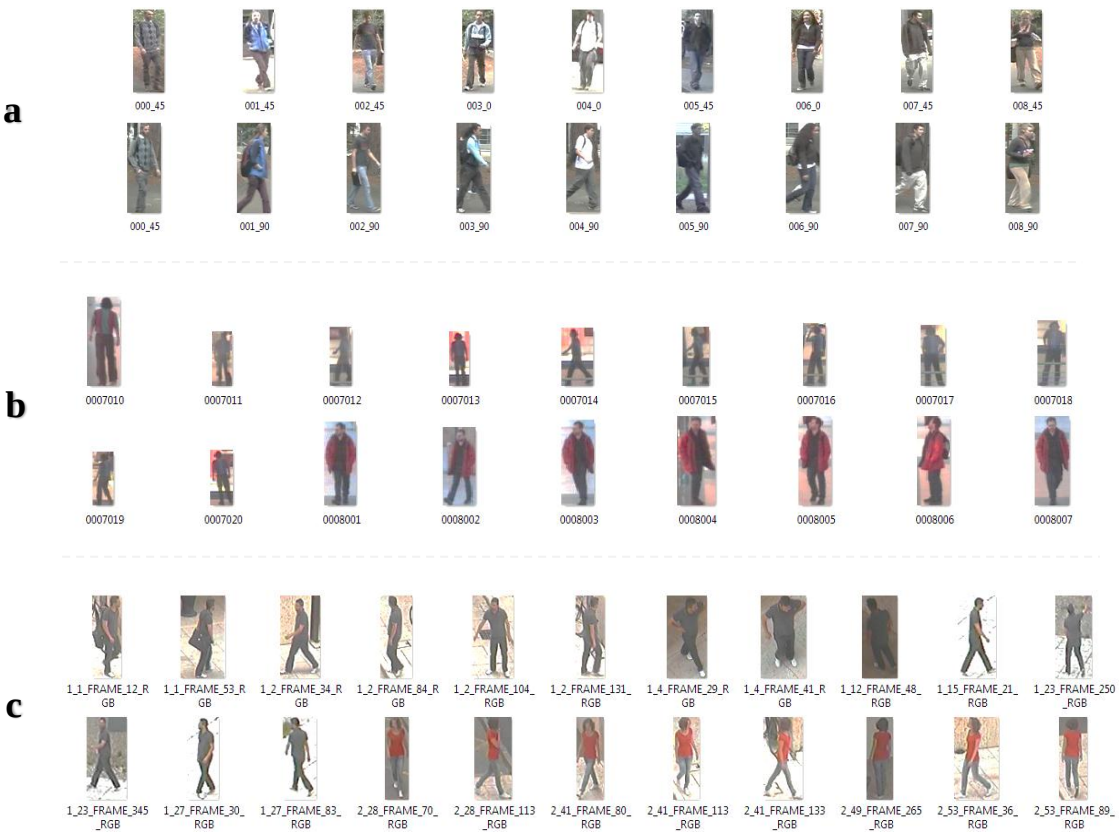

Figure 1.1: (a) Sample images from VIPeR dataset showing two images for every individual from two different viewpoints. (b) Sample images from CAVIAR dataset. (c) Sample images from 3DPes dataset.

Chapter 2

Appearance Model

2.1 Introduction

In a surveillance system, the detected person should be represented by a compact descriptor. The descriptor transforms an image patch from a matrix of intensities to a vector. Having a strong descriptor is a key factor of having a robust surveillance network. Retrieving images or searching the network for correspondences in reasonable time and accuracy are made possible by different image descriptors.

There is an efficiency requirement on the descriptor used for surveillance system because of the continuous change in resolution and view point as people move and orient.

Re-identifying people by their appearance is often a challenging task, especially in a cluttered environment. Since people move continuously, their appearance depends on the object orientation with respect to the camera.

In many applications, facial features are used to re-identify people. However, this approach requires that a face of enough resolution is visible in all cameras. This is usually not the case in surveillance systems. This means that the re-identification should be performed using other methods such as color and texture. For example in [24, 25]; color descriptors based on color histograms are used.

2.2 Colors

It was discovered that when a beam of sunlight passes through a glass prism, the beam of light is not white but it consists of a continuous spectrum of colors ranging from violet at one end to red at the other. The color spectrum is divided into six broad regions: violet, blue, green, yellow, orange and red [26].

Essentially, the colors that the human sees are determined by the nature of the light reflected from an object. A body that reflects light that is balanced in all visible wavelengths appears white to the viewer. In contrast, the body that favors partial range of the visible spectrum shows some color.

Generally, the characteristics used to differentiate one color from another are *brightness*, *hue* and *saturation*. Brightness symbolizes the achromatic representation of intensity. Hue is the attribute associated with the dominant wavelength in the mixture. Saturation is the amount of light mixed with the hue.

2.2.1 Color models

The color model is a specification of a coordinate system where each color is represented by a single point. The main purpose of it is to simplify the specification of colors in some standard [26]. The color models are usually oriented either toward applications where color manipulation is the aim of the model or toward the hardware like TV's and printers.

2.2.1.1 RGB color model

In this model, each color appears as a mixture of it primary component Red, Green and Blue. The model is based on the Cartesian coordinates system. The color space is a cube, as shown in Figure (2.1); in which the primary colors are on the three main axes. The other corners are the secondary colors Cyan, Magenta and Yellow. The point of origin is the Black whereas the White is the corner farthest from the origin. The Grayscale lives on the line connecting the black with the white.

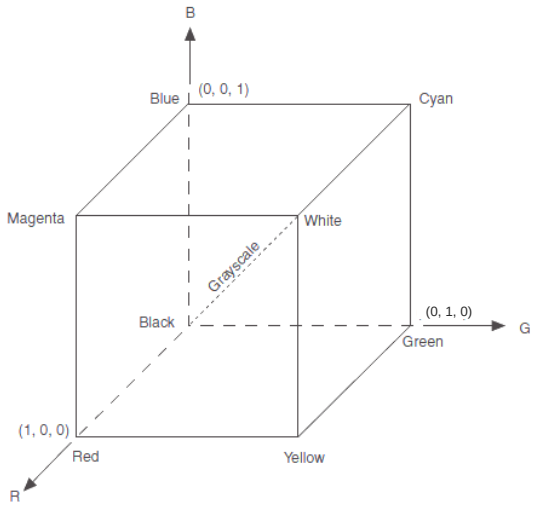

Figure 2.1: The RGB color model mapped to a cube. The horizontal x-axis as red; values increasing to the left, y-axis as Green, increasing to the lower right and the vertical z-axis as Blue, increasing towards the top. The origin, black, is the vertex hidden from view.

Images represented in the RGB color model consist of three components, each of which is an image holding the information for the primary color it represents as shown in Figure (2.2).

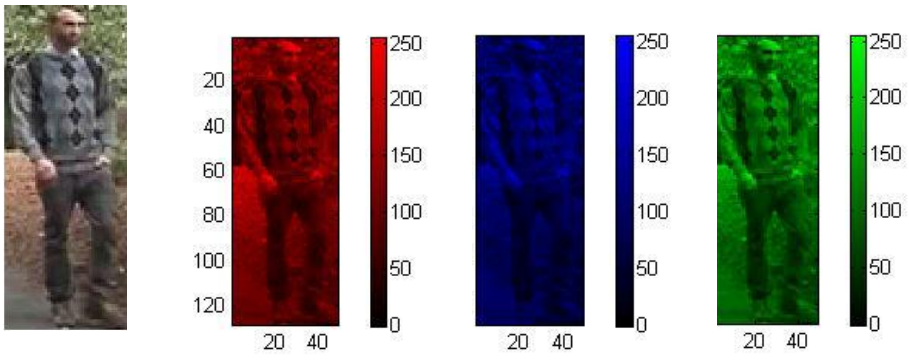

Figure 2.2: RGB color image components.

2.2.1.2 HSI color model

Unfortunately, RGB and similar color models are not appropriate for describing color for humans in terms that are practical for human interpretation [26]. Humans do not think of colored images as being composed of three layers that combine to form a single image. When humans see color objects, they describe it by its hue, saturation, and brightness. Hue is the color attribute of the dominate color while saturation describes its purity.

The HSI model decouples the intensity component from the color-carrying information [26] and this makes it the perfect tool for developing an image processing algorithm.

To understand the relation between HSI model and RGB, recall the RGB cube from Figure (2.2), stand this cube on the black vertex with the white vertex directly above it. See Figure (2.3).

To determine the intensity component for a specific color point, pass a plane containing the point and perpendicular to the intensity line. The intersection point is the intensity.

As mentioned earlier, the saturation is the purity of the color, so the farther the color from the intensity line the purer the color. The points that lay on the intensity line have a saturation of zero and this is why they appear to be gray.

In order to show how to compute the hue from a given RGB point, consider Figure (2.4).

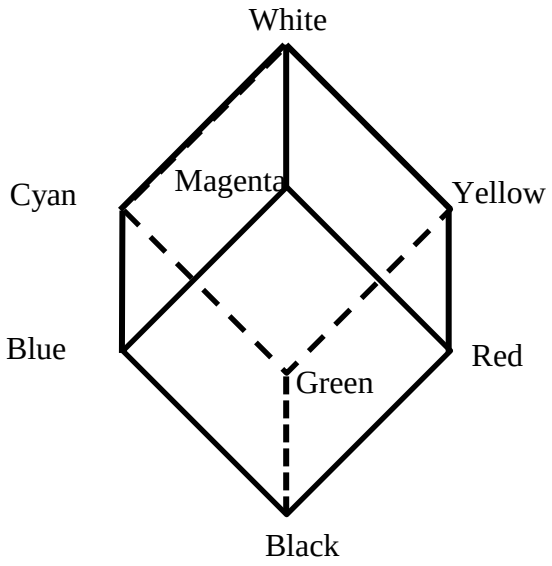

Figure 2.3: HSI cube, it is an RGB cube rotated to sit on its Black vertex with the White vertex on the top.

Connecting the White, the Black and any other RGB point forms a plane. This plane has the property of having the same hue for all the points contained in it. The reason is that the black and white colors cannot change the hue. It must be noticed that points inside the plane have different saturation and intensity. By rotating the shaded area around the intensity line, a different hue can be obtained.

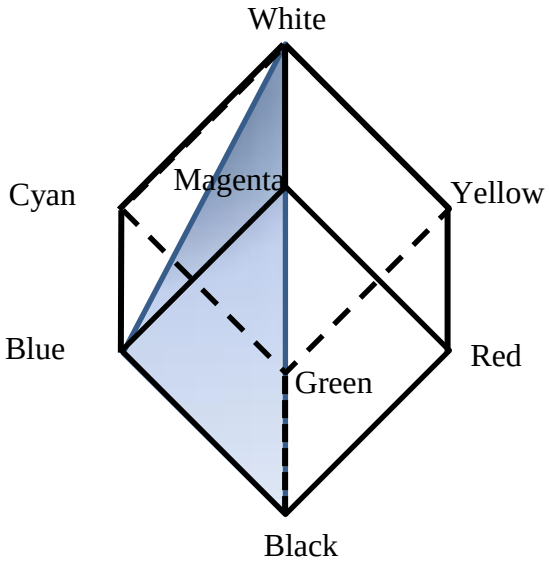

Figure 2.4: HSI cube with Hue plane shaded. Connecting the white point and the black point and any other RGB point forms a plane. This plane has the property of having the same Hue for all the points contained in it. The reason is that the black and white colors cannot change the hue.

2.2.1.3 YCbCr color model

The YCbCr is different from all the previous spaces. It is not an absolute color space; rather it is a way of encoding RGB information. Y is the brightness information (luma) while the Cb and Cr are the difference in red and blue respectively (chroma). This model separates the *luma* from the *chroma* component.

2.3 Color histogram

One of the key tools in physics-based vision has been color histogram [27]. Color histograms are a key factor in image understanding. Color is usually

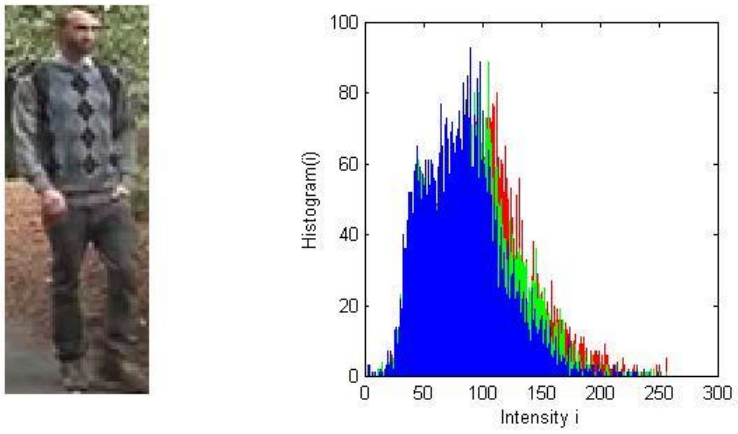

Figure 2.5: (Left) RGB sample image. (Right) Color histogram. As shown it has three histograms corresponding to the three layers of the RGB image.

thought of as an important property of an object. Color histogram is the distribution of pixels in a color space. In other words, it is the probability of the pixel being a particular color. See Figure (2.5).

A color histogram H for a given image is defined as a vector:

$$H = \{H[1], H[2], H[3], H[4], \dots, H[i], \dots, H[N]\}$$

Where (i) represents a color in the color histogram, $H[i]$ is the number of pixels of the color i in that image.

The use of the full range color histogram is usually not practical. It consumes memory, adds complexity to the next stage of calculations and is sensitive to minor changes in the image. To solve this problem, binning is used. It is simply dividing the histogram into a number of bins which is less than the full range number of bins. For example, if we have an 8-bit image we need 2^8=256 bins to cover the full color range as shown in Figure (2.5).

Using quantization, we can easily represent the same image using fewer bins. An example of a histogram with ten bins is shown in Figure (2.6)

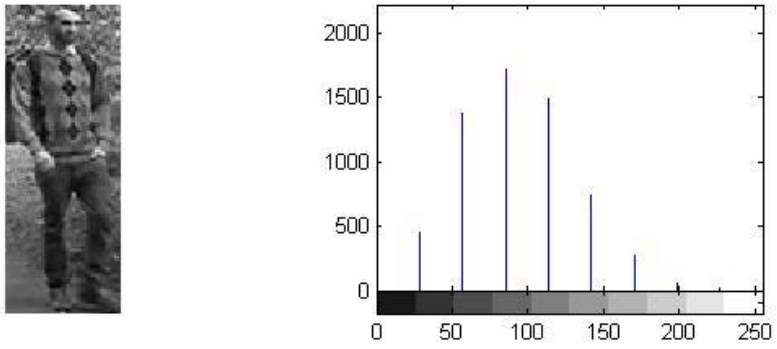

Figure 2.6: (Left) Gray scale sample image. (Right) Ten bins quantized histogram. This means that the pixels with values between 0-25.5 fall into the first Bin. The pixels with values between 25.6-51 fall into the second Bin and so on.

The choice of quantization factor is an issue need to be solved by balancing the following factors; if the number of bins is large, the computation complexity is high and the sensitivity of each bin in two images that belong to the same person

is high as well. If the number of bins is low, the histogram may lose some import information by merging slightly different colors into one bin.

There are two ways of representing images with the color histogram; global color histogram and local color histogram. Global color histogram means representing the image with a single histogram calculated over the entire image. It's relatively easy to calculate and store. However, it does not include information concerning the color distribution in different image regions. This is why the local color histogram is used. It is performed by dividing the image into sub-blocks then calculates the histogram for each of them.

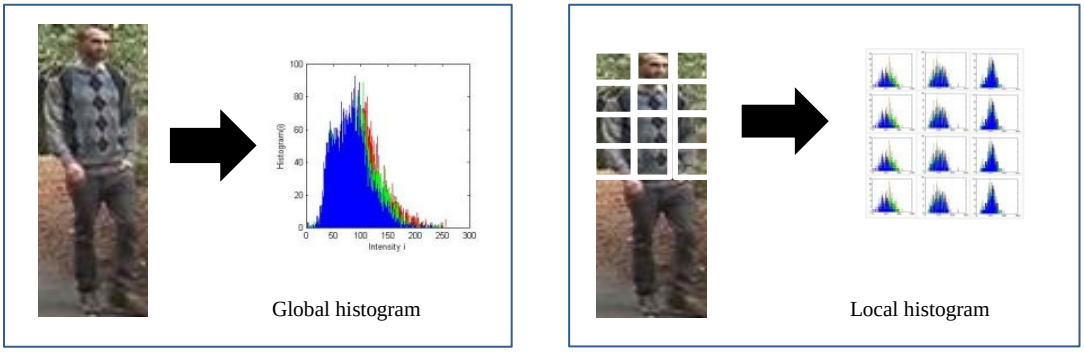

Figure 2.7: (Left) An image and its Global color histogram. (Right) Sampled image and the corresponding local color histograms.

2.4 Scale Invariant Feature Transform Descriptor (SIFT)

Descriptors are used to convert image patches to some other form that has desired properties. SIFT descriptor [28] is used to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variations, such as change in illumination or 3D viewpoint.

Once the keypoint is detected, the image gradient magnitudes and orientations are sampled around its location, using its scale to select the level of Gaussian blur for the image. These are illustrated with small arrows at each sample location on the left side of Figure (2.8).

Each sample point magnitude is assigned a weight calculated using a Gaussian weighting function with σ equal to one half of the width of the descriptor

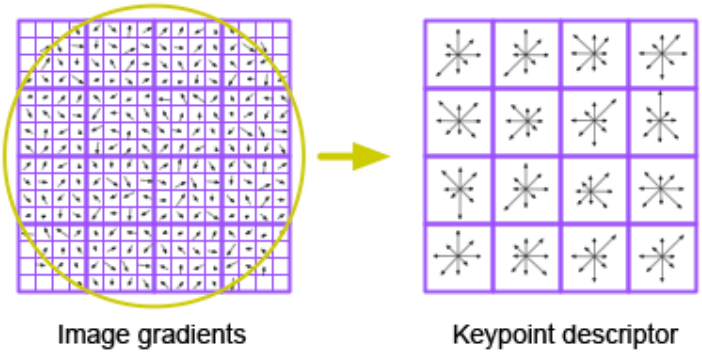

Figure 2.8: (Left) Gradients calculated over image patch. (Right) SIFT descriptor.

window. It is shown as circular window on the left side of Figure (2.8). The drive of this Gaussian window is to avoid rapid changes in the descriptor with small changes in the position of the window. It also gives less importance to gradients that are far from the center of the descriptor, as these are most affected by registration errors.

The descriptor is shown on the right side of Figure (2.8). It allows for a significant shift in gradient positions by creating orientation histograms over 4x4 sample regions. The descriptor shows eight directions for each orientation histogram. The length of each direction arrow corresponds to the magnitude of that histogram entry.

An important issue the descriptor should take care of is the boundary effects in which the descriptor changes sharply as a minor shift in the sample appears to happen. This could be a shift from one histogram to another or of one orientation to another. Therefore, each entry into a bin is multiplied by a weight of *1-d* for each dimension, where *d* is the distance of the sample from the central value of the bin as measured in units of the histogram bin spacing [28]. David Lowe mentioned in [28] that the best practical results are achieved with a 4x4 array of

histograms with 8 orientation bins in each. Therefore, the experiments in this paper use a 4x4x8 = 128 element feature vector for each keypoint.

In order to alleviate the effects of illumination, the feature vector is normalized to unit length. A change in contrast in which each pixel value is multiplied by a constant will multiply gradients by the same constant so the normalization takes care of this contrast change. On the other hand, the brightness in which a constant is added to the pixel value will not affect the gradient value as they are computed from pixel differences. However, non-linear illumination changes can also happen due to illumination changes that affect 3D surfaces with differing orientations by different amounts. These effects are less likely to change the gradient orientation but could affect the relative magnitude. Therefore, thresholding is used to reduce the impact of large gradient orientations. It is experimentally proven that a threshold of 0.2 works great [28].

2.5 Bag of words model:

It is a model that represents an image as a collection of local patches. Each patch is represented by a codeword from a large vocabulary of codewords [34]. The goal of learning is to achieve a model that best represents the distribution of these codewords in each category of scenes. In recognition, it first identifies all

the codewords in the unknown image. Then it finds the category model that fits best the distribution of the codewords of the particular image. Beli et al. came up with the idea of Latent Dirichlet Allocation (LDA) [29]. It is a model that can use a vocabulary of words from a document to explain its topic. In this model, the words are the only variable used to predict the topic of the document.

LDA has been applied in a number of computer vision areas. It is successfully used in image segmentation and labeling [30]. Given a large dataset of images, a number of methods are used to segment each image. These are treated as documents for LDA [3]. A histogram of visual words is then computed for each segment document. Then LDA model is trained in order to discover the topic of the set of documents.

Another application is discovering the natural scene categories from a set of images. In [34], a modified LDA is presented. The process starts by choosing a category label; for instance 'forest'. Given the forest class, the distribution vector is drawn and this will determine what themes to select while generating each patch of the scene. To build a scene, a theme is sampled. Use the theme 'tree' as an example. Based on the sampled theme, a codeword is drawn before going back to sampling. There are different methods to extract patches: evenly sampled grid,

random sampling, salient regions detected by Kadir & Brady detector [35] or SIFT descriptors [28]. Once categories are learned, the machine can be used to classify new incoming images. The category label corresponding to the highest likelihood probability is assigned to each test image.

In this thesis, the bag of word model is used to build a signature for every image. We generated two code books; one that captures the color features while another captures the texture features.

2.6 Clustering

Clustering, in definition, is the task of allocating groups to a set of entities. These entities that belong to the same group have, in some sense or another, similarities. The groups are usually called '*Clusters*'. Clustering has a wide range of usages. It is used in data mining. It is so commonly used in the fields of machine learning, information retrieval, and image analysis and pattern recognition. The word *clustering* refers to the task and not to a specific algorithm. There are many various algorithms to approach the task of clustering. They usually differ significantly from each other in many aspects; the notion of what constitutes a cluster and the efficiency in finding them. One of the popular

methods used in deciding which object belongs to which cluster is based on the distance between the objects in the data space.

Another method is based on a particular statistical distribution. Dense areas can be used as an indication to form clusters. To use a clustering algorithm, one should keep in mind that there is no general setting that can be used for every task. Parameters such as the number of expected clusters, the distance function used, and the density threshold are dataset dependent. In addition, the intended use of the results may have an impact on how to define those parameters.

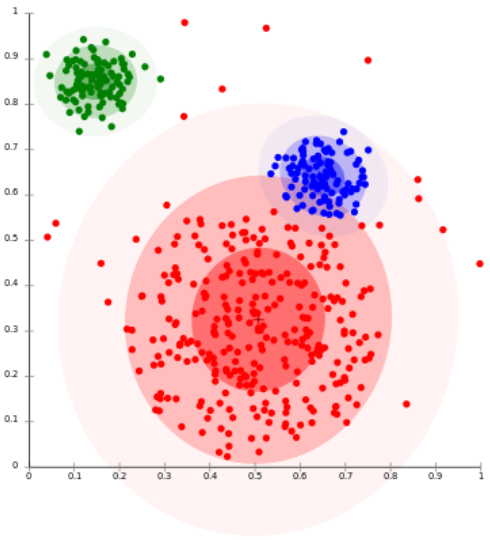

Figure 2.9: Clustering is the task of allocating groups to a set of entities.

The process of clustering is far from being automated; instead it is an iterative procedure of multi-objective optimization that could involve trial and error. In

addition, it sometimes has to be done with a reverse technique; meaning the preprocessing parameters will change until acquiring the desired properties.

2.6.1 k-means clustering

One technique that can be used to simplify computation and speed up convergence is *k*-means clustering algorithm. Although it is very simple, it is a popular approximate method. The goal of *k*-means is to group *n* observations into *k* clusters, in which each observation belongs to the cluster with the nearest mean vector.

The proper number of clusters needs to be decided before clustering. This number, as mentioned earlier, is problem dependent [31]. For instance, to categorize a dataset of English alphabetical images, *k* is assigned to be 26, which is logical.

Given a set of observations $(x1, x2, x3, \ldots., xn)$ where each observation is a vector of *d* dimensions, *k*-means aims to gather these observations into *k* clusters, each of which is represented by its mean μ_i where$(i = 1 \rightarrow k \; and \; k < n)$. The objective function that *k*-means tries to minimize is as follow:

$$\arg\min \sum_{i=1}^{k} \sum_{xj \in Ci} \|x_j - \mu_i\|^2$$

Where C_i: *indicates the cluster*. i: *cluster number* $= 1 \rightarrow k$. μ: *cluster mean*.

To summarize the algorithm: see Figure (2.10).

Algorithm 1 *(k-means clustering)*

1 begin initialize $n, c, \mu_1, \mu_2, \ldots, \mu_c$

2 do classify n samples according to nearest μ_i

3 recomputed μ_i

4 until no change in μ_i

5 return $\mu_1, \mu_2, \ldots, \mu_c$

6 end

The computational complexity of the algorithm is $O(ndcT)$ where d is the number of features and T is the number of iterations.

K-means is used in this thesis to group the codebook into a predefined number of clusters. The centers of the clusters help with the building of the 2D image signature.

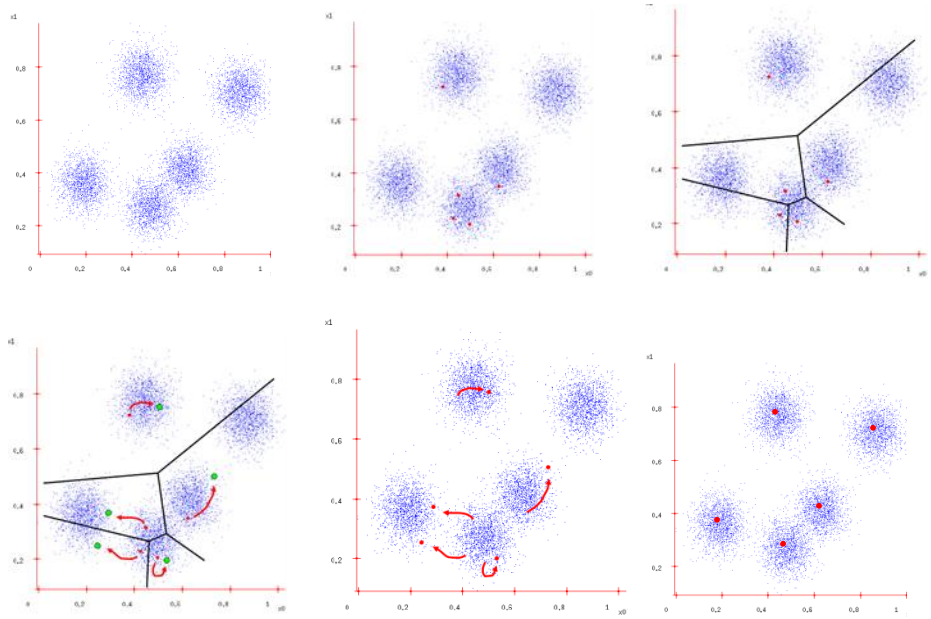

Figure 2.10: k-means clustering steps; it starts by initializing k centers randomly. The next step is to classify the data points according to the nearest center then recalculate the centers as the mean of the data groups.

2.7 Adaptive Boosting (AdaBoost)

Boosting is a general method for improving the accuracy of any given machine learning algorithm [33]. Machine learning studies the techniques that make accurate predictions based on past observations. For example, suppose that we have an article and we would like to decide whether it is political or sport related. After choosing a machine learning algorithm, we feed this algorithm with

articles and their labels (political, sport related). The algorithm learns a prediction rule. When given a new article, the rule attempts to predict which category it belongs to.

Building a rule that is able to give an accurate prediction is not an easy task. On the other hand, it is not hard to come up with a rule of thumb that is only moderately accurate. We are looking for a rule that is able to give a prediction that is slightly better than random guessing. An example of such a rule is if the word 'football' occurs in the article, then it is a sport related article; it says nothing about what to predict if the word football doesn't occur. Certainly, such a rule gives better predictions than random guessing.

Boosting is based on the observation that learning many rough rules is easier than learning one highly accurate prediction [33]. To apply boosting, the 'weak' or 'base' learning algorithm is called repeatedly; each time with a different distribution of training examples. Each time it is called, the learning algorithm produces a weak prediction rule. After many rounds, the boosting algorithm must combine these weak rules into one single prediction rule that will be much more accurate than any of the weak rules.

2.7.1 AdaBoost Algorithm

There are number of variations on basic boosting. The most popular is AdaBoost, which stands for Adaptive Boosting. This kind of boosting is the most flexible; allowing the designer to add weak learners until some desired low training error has been reached. Each training sample receives a weight that decides its probability of being selected for a training set of an individual component classifier [31]. Being correctly classified, the chance of the pattern being used again in a subsequent component classifier is reduced. Conversely, if the pattern is not correctly classified, its chance of being selected again is raised. In this way AdaBoost gives more importance to the hard examples and focuses on getting their labels right. The algorithm starts with initializing the weights for all the patterns to be uniform. On each iteration *k*, we train the component classifier, *Ck,* which best classifies all of the patterns according to the current weights. Then the weights of the samples that are misclassified are increased by *Ck,* while the weights of the patterns that are correctly classified are decreased by *Ck.* Patterns are chosen according to this new distribution which is used to train the next classifier, *Ck+1,* and the process iterated.

Let *xi* denote the patterns with labels *yi*, and let *Wk(i)* be the *kth* discrete

distribution over all these training samples. The AdaBoost procedure is as follows:

Algorithm 2 *(AdaBoost)*

1 begin initialize $D = \{x_1,\ y_1, x_2,\ y_2, \ldots, x_n,\ y_n,\}, W_1(i) = \frac{1}{n}\ , k_{max}$
2 $k \leftarrow 0$
3 do $k \leftarrow k + 1$
4 Train weak learning C_k using D sampled according to distribution $W_k(i)$
5 $E_k \leftarrow$ Training error of C_k measured on D using $W_k(i)$
6 $\alpha_k \leftarrow \frac{1}{2}\ln[\frac{1-E_k}{E_k}]$
7 $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} * \begin{cases} e^{-\alpha_k}\ if\ h_k(x_i) = y_i\ (correctly\ classified) \\ e^{\alpha_k}\ \ if\ h_k(x_i) \neq y_i\ (incorrectly\ classified) \end{cases}$
8 until $k = k_{max}$
9 return C_k *and* α_k *for* $k = 1$ *to* k_{max} *(ensemble of classifiers with weights)*
10 end

It can be easily noted that at line 5 the error for classifier *Ck* is calculated with respect to the distribution, *Wk(I),* on which it was trained. *Z*, in line 7, is a normalization factor that keeps the distribution *W* true. *Hk(xi)* is the output label (+1 or -1) of the classifier *Ck* given the pattern *xi.*

We terminate the loop upon reaching a particular error rate.

Given a test point, the decision is simply made based on the discernment function. It is the weighted sum given the component classifier:

$$g(x) = \left[\sum_{k=1}^{kmax} \alpha_k h_k(x) \right]$$

The classifier output is the sign of the g(x).

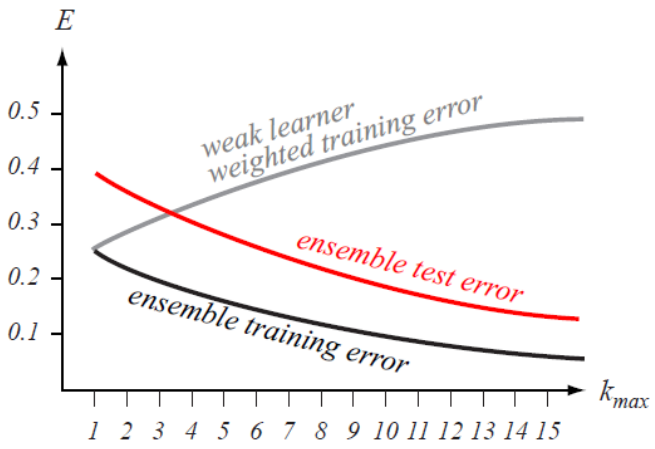

Figure 2.11: AdaBoost applied to a weak learning system can reduce the training error E exponentially as the number of component classifiers, kmax, is increased. Because AdaBoost focuses on difficult training patterns, the training error of each successive component classifier is generally larger than that of any previous component classifier (shown in gray). Nevertheless, as long as the component classifiers perform better than chance, the weighted ensemble decision insures that the training error decreases. It is often found that the test error decreases in boosted systems as well, as shown in red.

The total error of the ensemble can be made low by setting the number of component classifiers, *Kmax,* sufficiently high. To illustrate, notice that the training error of the weak learner, *Ck,* can be written as $Ek = \frac{1}{2} - Gk$ for some positive value *Gk* thus the ensemble training error is:

$$E = \prod_{k=1}^{kmax} \left[2\sqrt{E_k(1-E_k)}\right] = \prod_{k=1}^{kmax} \sqrt{1-4G_k^2} \leq exp^{\left(-2\sum_{k=1}^{kmax} G_k^2\right)}$$

As shown in Figure (2.11) it is sometimes helpful to increase *Kmax* outside the value needed for zero ensemble training error because this may improve generalization.

The AdaBoost makes use of two types of decision models, decision trees and decision stumps. Decision stumps are simply a one level decision tree. In other words, a decision tree with only one root node immediately connects to two terminal nodes. On the other hand, the decision trees are multilevel decision nodes. Each leaf node may have branches.

Generally, the longer you train the model the more it overfits the training data, which usually applies to most of the machine learning algorithms. Fortunately, it does not apply to AdaBoost.

Looking at the AdaBoost behavior, it appears that boosting violates the no free lunch theorem in that the ensemble classifier always performs better than any of its single components. According to the error equation above, the training error drops exponentially with the number of component classifiers. The theorem is not violated, however, boosting only improves performance if the component classifier

performs better than chance which is impossible to guarantee a priori. Additionally, the reduction of error on the training set does not ensure reduction of the testing error; the generalization.

As shown above, AdaBoost needs a collection of week classifiers to choose from. Once the weak classifiers are ready, a model can be trained to provide maximum separation between the positive and negative examples. In this thesis, the week classifiers are the features themselves. We used a four nodes tree as a weak classifier that can separate the training samples into a positive and a negative. Each feature is an individual weak classifier. It has its own weight in the final model and its associated threshold.

After extracting all the features from the training data, we let AdaBoost loop around all of the features to find the best one. The one that can separate the data with minimum error regarding the current distribution is best. We then add this feature to the ensemble of the model with its weight and the tree. AdaBoost goes through all of the features until reaching a pre-defined error or a particular number of iterations.

When given a testing pair, the feature vector is extracted then applied to the model to get the sign for the pair which is what we need.

Chapter 3

People Re-Identification in a Camera Network

In this chapter, we propose an algorithm to search video clips that shows a person of interest in a distributed camera network. This assumes that the tracking system is already built and providing images of people crossing the surveillance area. In this system, an analyst may identify the target person in one image from one of the network cameras, and run a query of "What other cameras show the same person?" or another scenario could be, when a new person crosses one of the network cameras, the system registers this new person with an identifier that can be recognized every time that person is noticed in the network.

Both scenarios can closely track the entire route taken by the target within the surveillance area.

3.1 Overview

As mentioned earlier, we needed a model that captures the color and texture properties of a person. Such a model will need to be able to tolerate the changes in illumination. We formulated a model that uses a mixture of color histograms and SIFT [28] feature as its basic elements, keeping in mind a set of constraints that are related to temporal and appearance consistency of the people.

In this work, we assume the cameras collect data independently and store the data of the detected people in a database. Our model builds upon on an already existing detection and tracking system; assuming that software already exists with the ability to perform these two tasks. Such a detection and tracking system feeds our model with people images.

The focus of this chapter is the formulation of an image signature and the training of the model. The input of the system is a set of images of detected people coming from the camera network. The network can be of any setup.

Figure (3.1) shows a sample setup [17] of eight cameras covering part of university campus.

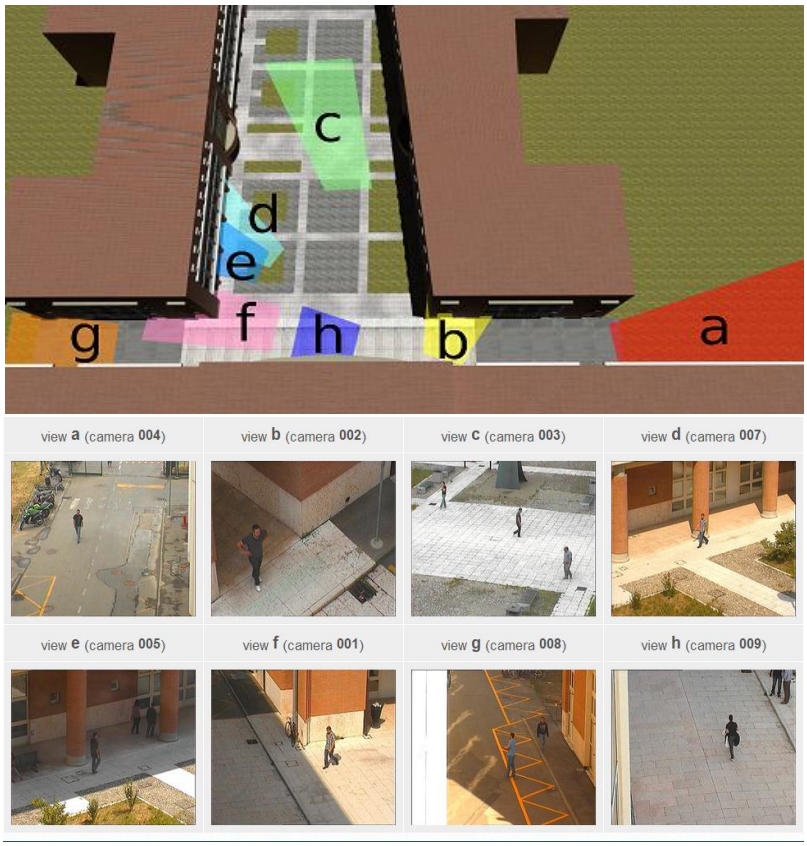

Figure 3.1: Sample setup of eight cameras covering part of university campus. The image is a part of 3DPes: 3D People Dataset for Surveillance and Forensics project.

To describe the problem of detection and tracking, video streams are scanned (online or offline) for the existance of people. Each time a person is detected, the tracking system starts capturing images of that person, and it stores them in dataset that builds up a stack of images, See Figure (3.2). Having multiple

cameras in the network, we acquire a database of images coming from different locations that can be of the same or different individuals, see Figure (3.3).

Figure 3.2: Stack of images captured by an image detector. Once the system detects a person, a stack of images are captured and stored in a database.

The result of using the tracking system is shown in Figure (3.3). From this point our model takes over. The building blocks of the model are the images coming from the cameras. Such images, as shown in Figure (3.3) may suffer from severe illumination changes.

Figure 3.3: Images from different cameras in a camera network showing two individuals from different point of view.

3.2 Color Histogram – SIFT model:

Our appearance model is basically a hybrid of a texture and histograms. It is empowered by the simplicity of the feature design, so there is no complicated feature scheme. To get the task done, a machine learning algorithm took over the part of selecting what is good in the simple feature and left out what is not useful.

To illustrate the general frame work, after the people are detected, tracked and camera views are stacked, the process of features extraction is ready to begin. We will explain this process thoroughly in section (3.3). The output of the

feature extraction phase is a set of feature vectors corresponding to the input images. Each and every image has its own high dimensional feature vector.

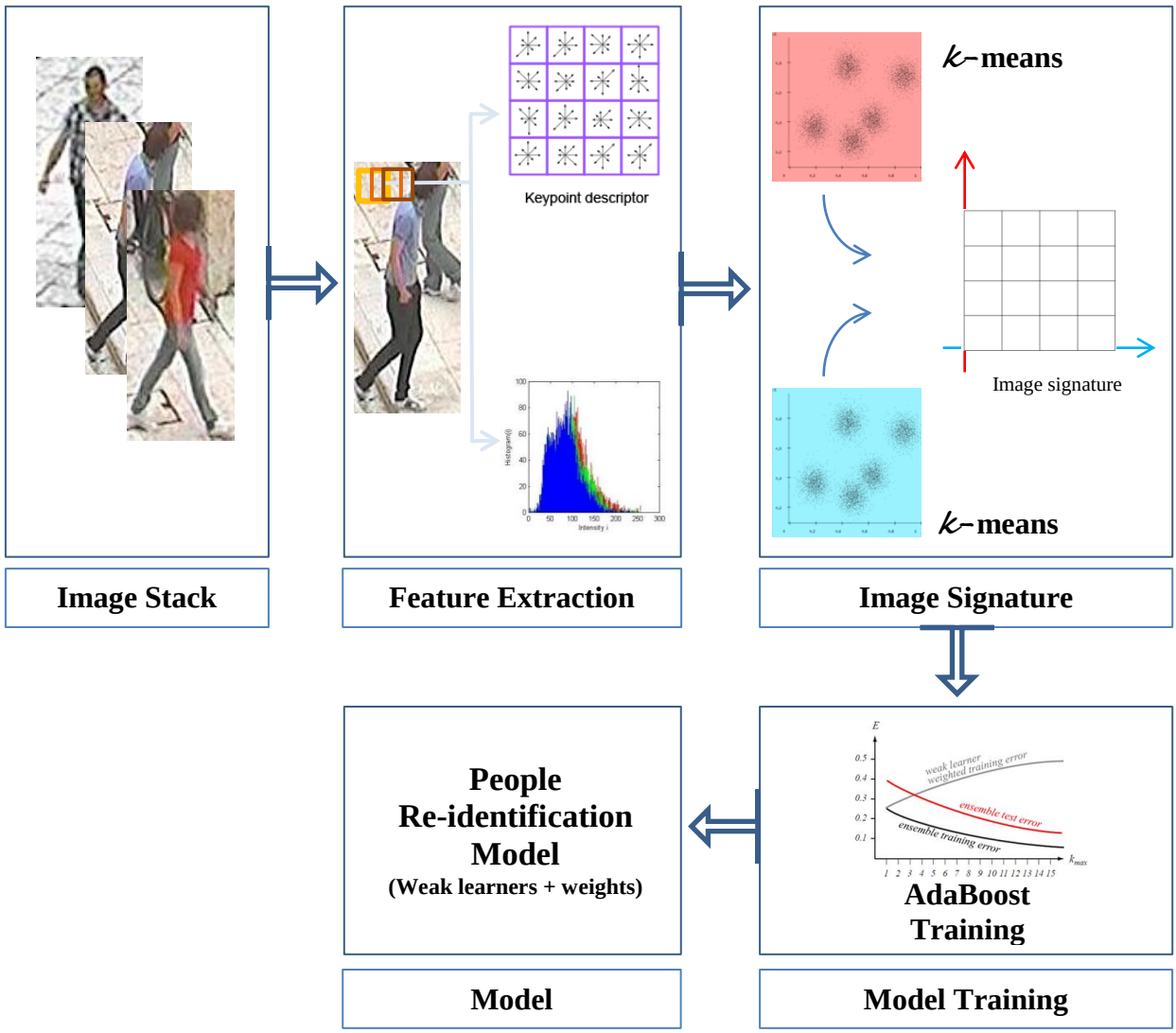

Figure 3.4: SIFT-HIST model. Every image is intensively sampled and features are extracted for every sample. Bag-of-word model is then used to create pools of patches, one for each feature kind. *k*-means is used to group the patches into a number of centers. Finally the centers are used to build a two dimensional template that facilitates the form of an image signature. AdaBoost is used to train the model.

In order to group the high dimensional feature vectors, including both the histograms and the SIFT descriptor features, *k*-means is used to find the centers for the distributions. These centers live on the two axes of the image signature. We call the two axes with the centers displayed on them a *signature map*. All the details are discussed later in this chapter.

The next step is to pair the images in order to form positive and negative examples. The example is a pair of images, hence, the difference between the two images feature vectors. When the two images belong to the same person, the resulted example is positive. If they belong to different individuals, the resulted example is negative. Once we have all the examples ready, we need a machine learning algorithm to select the features that provide the required separation between the positive and the negative pairs.

Having the model ready means we are ready to test incoming images. Each testing image goes through the same process of feature extraction, and then builds their signatures using the pre-set signature map from the previous training stage. To re-identify a person we pair the image we have with all the images of people who were in the network during a certain period of time and expose the pairs to the trained model to acquire positive correspondences.

Due to an imperfect model, as well as noisy and low resolution images, we expect the results to be imperfect too. The model could misclassify a positive pair to be negative or vice versa but in a tolerable frequency. To evaluate the model we used accuracy, recall and precision.

3.3 Features Extraction

Two types of features are extracted from each image, color and SIFT features. We start the feature extraction by densely sampling the image, see Figure (3.5). The size of the window and the overlap between two adjacent windows are determined by experimenting. Each image patch is processed independently and all the features are extracted.

Figure 3.5: Intensely sample the images by passing sliding window all over the image with a pre-specified overlapping ratio.

3.3.1 Color Features

Color features consist of nine channels RGB, HSI and YCbCr, each of which contributes to building the final signature of the image. Only one of the luminance (Y and I) channels is used. All eight channels are extracted from each image before going through the process of sampling.

A histogram is calculated for every image patch from every channel. Binning is used when calculating histograms so instead of using the full intensity range (0-255) we only use N bins while N<<255.

The process starts by sampling every color channel into patches with a specific size. The patches then go through a histogram extraction stage. Before calculating the histograms, the number of bins should be fixed. This number will not be changed through the entire process. The resulted histogram becomes a part of a long feature vector of the image where the patch came from. This process is repeated for all color channels from that patch. The resulted histograms are concatenated together and the outcome is a long feature vector corresponding to one image patch. This feature vector then joins a pool of feature vectors coming from different patches belonging to the same and to different images.

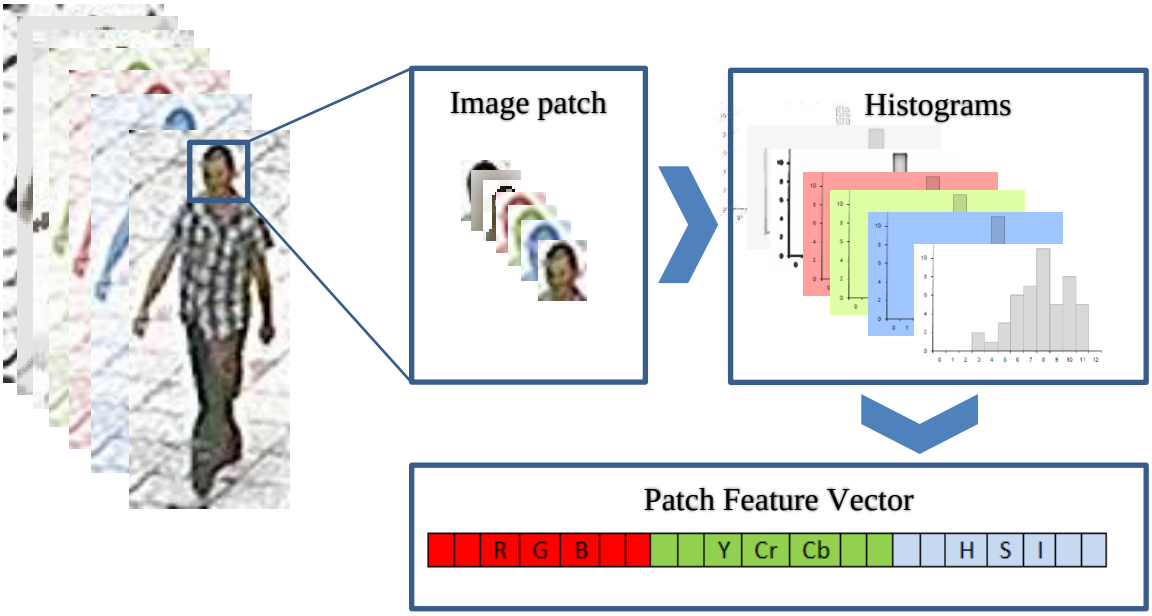

Figure 3.6: Process of color histograms feature vector extraction. Patches are extracted then histograms are calculated for every patch. The resulted histograms concatenate to form the feature vector.

Extraction of the color features from all the training data results a pool of patches. This pool is used to form the first dimension of the final two-dimensional image signature. The patches are simply points in the high dimensional space. We use *k*-means to find a specific number of centers in the space of patches. The number of centers will be selected ahead. *k*-means is then applied to the pool of feature vectors in order to find the centers of the distributions. These centers will be used later in building the image signature.

3.3.2 SIFT Descriptor

All image patches that are extracted go through SIFT feature extraction in order to form the second dimension of the image signature. It is performed using SIFT descriptor.

After extracting image patches, SIFT descriptor is calculated for every patch.

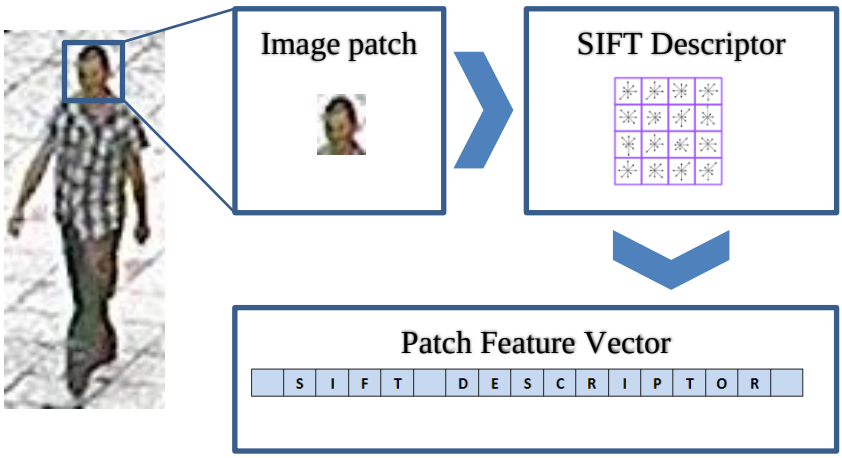

Figure 3.7: Process of SIFT feature vector extraction. Patches are extracted then SIFT descriptors are calculated for every patch.

The result is a 128 dimensional feature vector. The number 128 is calculated from a four-by-four descriptor with five orientations on each cell (4*4*5=128).

All the patches from all the training images are collected in a pool. k-means is applied to this pool of feature vectors with a pre-defined number of centers. This forms the second dimension of the image signature.

3.4 Image Signature

The final step in creating an image feature vector is to create an image signature. In this thesis we came up with the idea of two-dimensional image signatures. In order to overcome the problem of two images having similar color contents, another dimension is added to increase the signature reliability.

As we discussed earlier, we used the training data to create a template of a two dimensional matrix. The number of matrix cells along the x and y dimensions are to be decided ahead. The number of cells along the x axis is the number of k centers used to group the patches in the color histogram pool. The other dimension uses a number of k centers that are called on the SIFT feature pool.

Once we have the template ready, we let the algorithm scan all the training images. For every image, all the patches are scanned and every patch is assigned to the group with the nearest center. The patch will be assigned two numbers; one is the center from the color histogram groups and the other is from SIFT descriptor groups. Those two numbers guarantee a place into the two dimensional signature map. Once all the patches of one image are placed on the signature map, we fill every cell with the count of the patches placed on it. This matrix is then converted to a one dimensional array which is the final feature

vector of the image. To recall the steps of feature vector extraction, we first sample the image and extract two kinds of feature, colors and SIFT from each

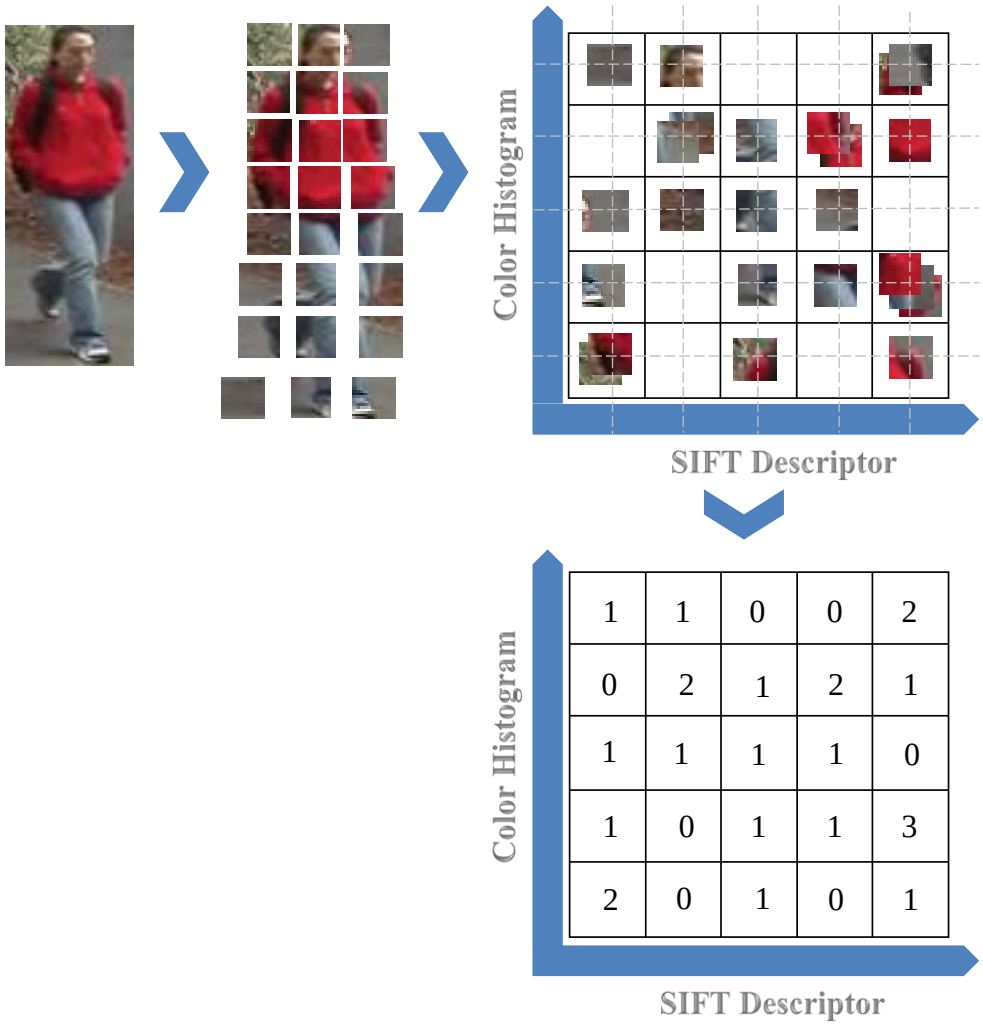

Figure 3.8: Image signature creation. Images are sampled intensively. The calculated signature pattern is used to accommodate the patches.

sample. K-means is used to group the color and SIFT feature pools. The group centers are used to create the two dimensional signature map which is used to form the feature vectors for all the images.

3.5 Model Training

Building a model that is able to separate positive examples and negative examples is the next step in our project. In order to let the machine decide what parts of our feature vector are good for the task and give them a higher importance, we chose to use AdaBoost algorithm. It fits perfectly with the way we formulated our design.

Given a feature vector for every training image, we need to formulate positive and negative examples. One example is a pair of images. The positive example means that the two images belong to the same person while the negative example means that the two images belong to two different individuals. *Delta* is used as an indicator to examples; it is the absolute difference between two features in the feature vector:

$$\delta = \left| v^{(1)} - v^{(2)} \right|$$

Where $v^{(1)}$ is a feature from one image in the pair and $v^{(2)}$ is the corresponding feature from the second image.

It may be noticed that the number of positive examples increases linearly while the negative examples increase exponentially. The use of all the negative examples leads to over fitting. The model loses the ability to recognize positive

examples. This problem is solved by down-sampling the negative examples so we can have balanced data.

Training starts by exposing AdaBoost to the *Delta* matrix given the labels as +1 for the positive examples and -1 for the negative examples. AdaBoost then iterates T times over the training data and finally produces a strong classifier.

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

Where x is the input example, h is the weak classifier at iteration T and α is the weight of the weak classifier.

3.6 Results

Out of all the data available, training a model for people re-identification requires multiple images of the same individual. We have chosen to use three datasets (VIPer, CAVIAR and 3DPes) for this reason. VIPeR dataset [21] contains two views of 632 pedestrians. CAVIAR [22] dataset contains 1220 images for 72 individuals. 3DPes [23] contains 1012 snapshots of 200 people.

3.6.1 Testing Evaluation:

In order to evaluate the algorithm, we used a 5-fold cross validation. Cross-validation is a method where the data is divided into number of folds (partitions)

performing the training on one subset (4 partitions), and validating the model on the other subset. To reduce inconsistency, multiple rounds of cross-validation are performed using different partitions. The validation results are averaged over the rounds.

As shown in Figure (3.9) and (3.10), the testing error drops with every iteration. The dashed blue line represents the *error rate* which is calculated as the ratio between the incorrectly classified examples to the total number of testing examples. On the other hand we can see that the *accuracy* (solid blue line) which is calculated as the ratio between the correctly classified examples to the total number of examples.

The dashed green line represents the *Recall* which is calculated as the ratio between the correctly classified positive examples to the total number of positive examples. *Precision* (dashed red line) is calculated as the ratio between the correctly classified examples to the total number of examples that are classified as positive.

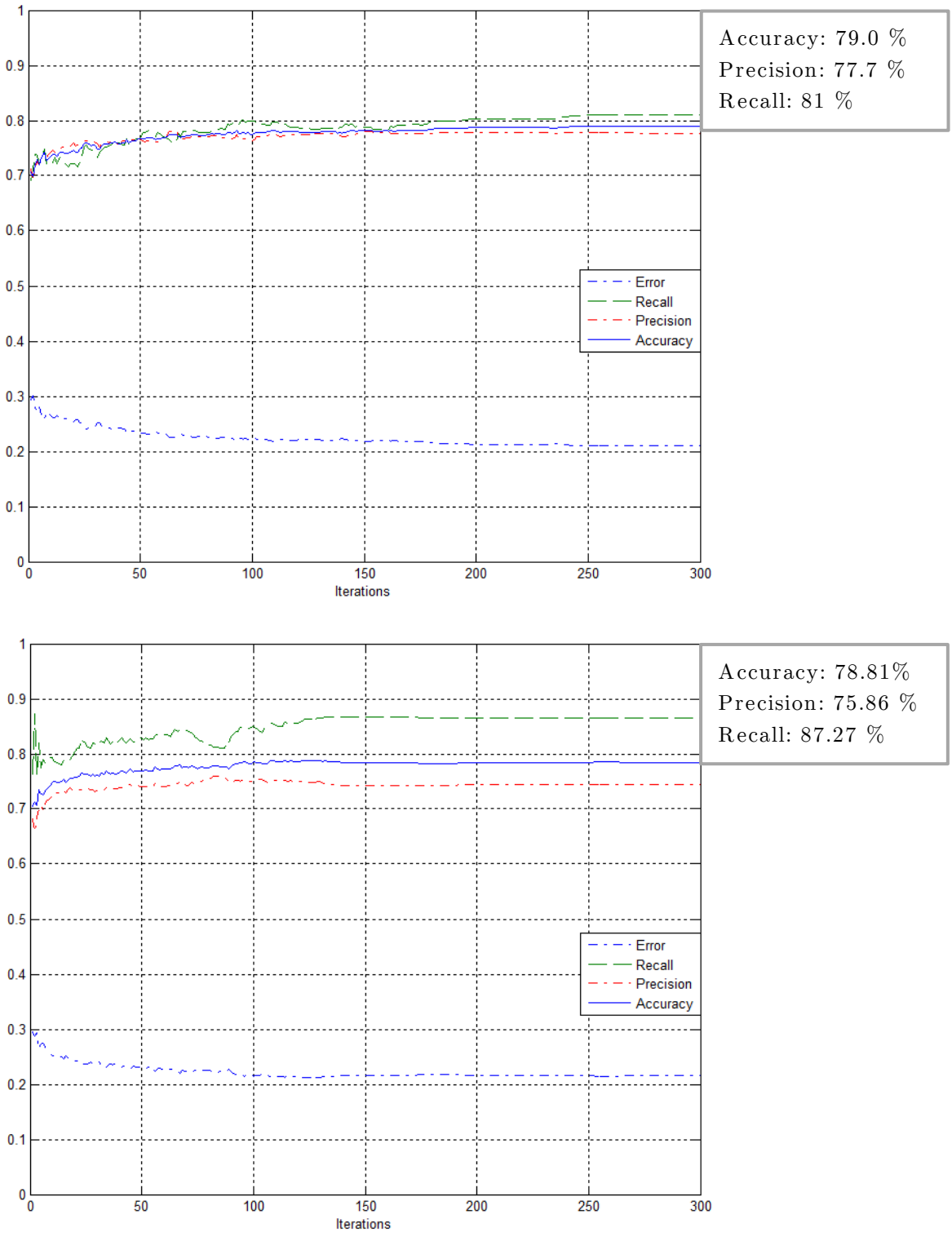

Figure 3.9: (Top) VIPer dataset. (Bottom) CAVIAR dataset. Testing Evaluation, the dotted Blue line shows the error decreasing as the iteration number increases.

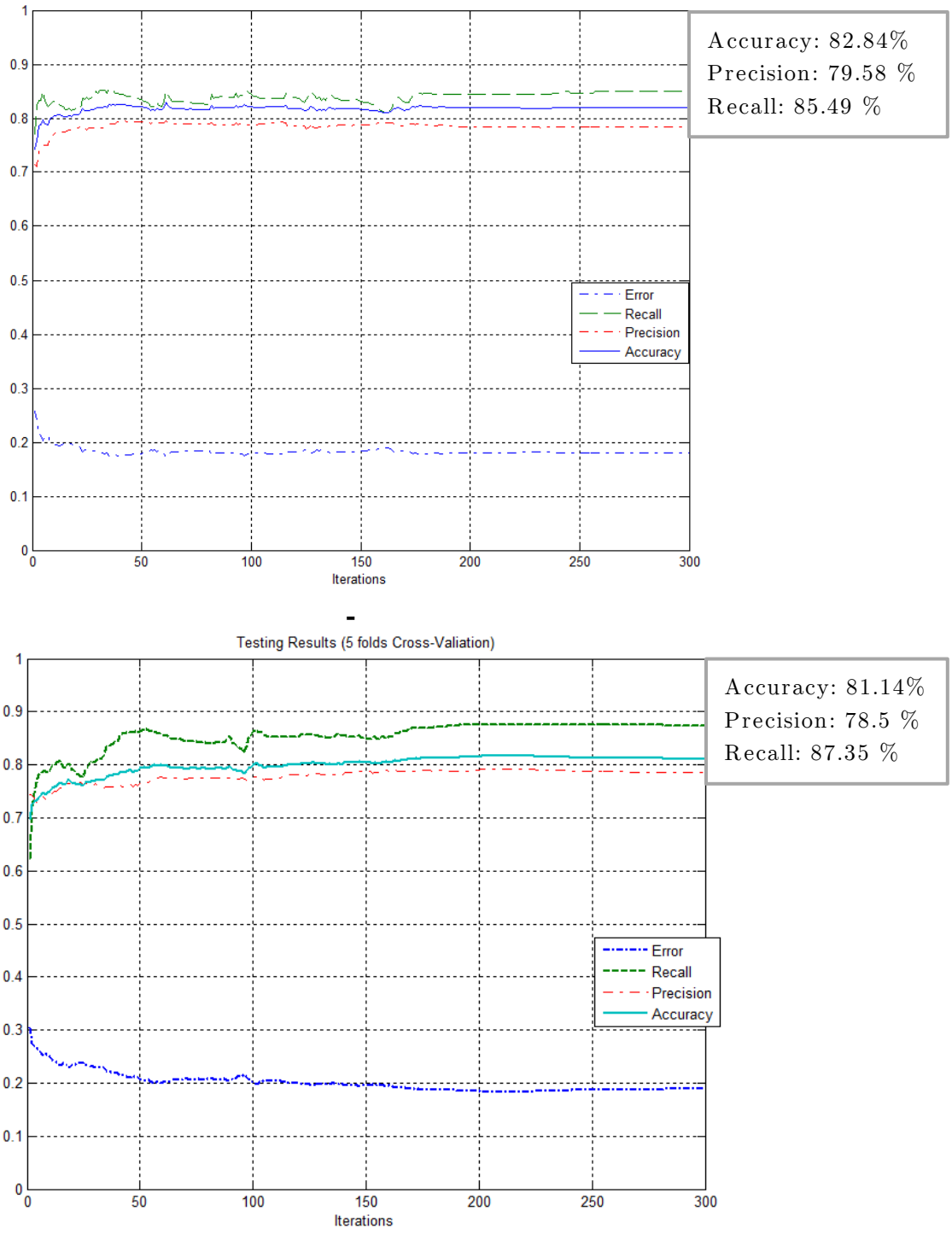

Figure 3.10: (Top) 3DPes dataset (Bottom) Overall performance. Testing Evaluation, the dotted Blue line shows the error decreasing as the iteration number increases.

3.6.2 Image Query

In this part of the testing we use random images as targets and the let algorithm give back all possible positive correspondences. See Figure (3.11)

Figure 3.11: (a) Target images. (b) Correctly retrieved images. (c) Incorrectly retrieved images.

In a re-identification system, the more targets, the harder the task of re-identifying individuals. Figure (3.12) shows how the performance of the system degrades as the number of targets increases. When the number of targets is very small, the performance is very high.

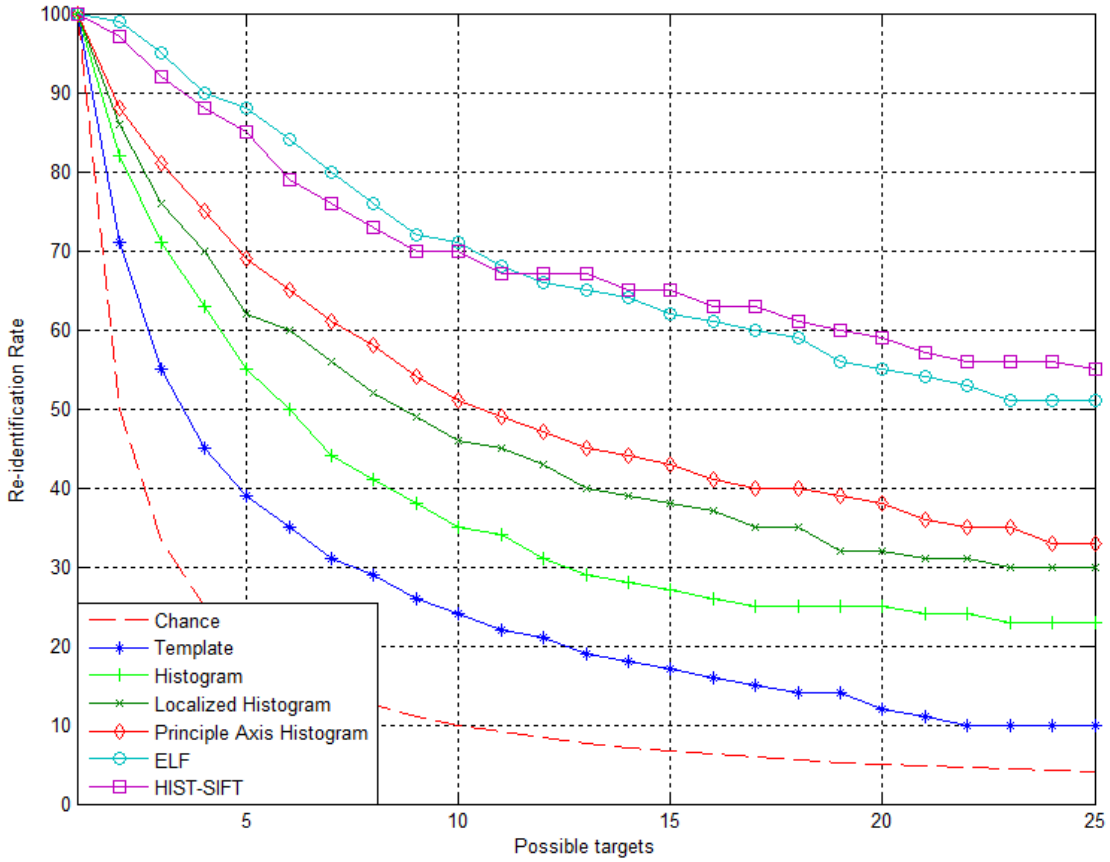

Figure 3.12: Re-identification rates vs. number of targets for benchmarks and our algorithm.

As notice from Figure (3.12), our algorithm is outperformed by the state of the art when the number of targets is relatively small but the state of the art degrades faster when the number of targets goes beyond ten.

Chapter 4

Conclusion

In this thesis we have discussed some important factors that should be considered while designing a re-identification system. Then we presented a method for re-identifying people in a camera network by using two-dimensional signatures.

We discussed the main techniques that can be used to describe people and re-identify them. The focus of this thesis was on the appearance features and descriptors. A mixture of color features and texture descriptors are used to build signatures for images.

We discussed a method that highlights the color and texture features of the image. Every image is intensively sampled and features are extracted for every sample. Bag-of-word model then used to create pools of patches, one for each

feature kind. k-means is used to group the patches into a number of centers. Finally the centers are used to build a two-dimensional signature map that facilitates the form of the image signature.

AdaBoost is used to create the model that is used to re-identify people in the camera network. AdaBoost is exposed to the features coming from images and trained to separate the positive examples from the negatives.

The system we produced is people orient independent. The use of AdaBoost makes it fast. The integration of HSI color model helped to make the model illumination independent too.

Bibliography

[1] Wei-Shi Zheng, Shaogang Gong, and Tao Xiang, "Person Re-identification by Probabilistic Relative Distance Comparison", IEEE Conf. on Computer Vision and Pattern Recognition, Colorado Springs, USA, June, 2011.

[2] D. Gray and H. Tao. Viewpoint invariant pedestrian recognition with an ensamble of localized features. In ECCV, pages 262–275, 2008.

[3] Lo Presti L., Sclaroff S., La Cascia M., "Path Modeling and Retrieval in Distributed Video Surveillance Databases," IEEE Transactions on Multimedia (TMM), 2012.

[4] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. “Person re-identification by symmetry-driven accumulation of local features”, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2010.

[5] M. Parizeau M. Lantagne and R. Bergevin. Vip : Vision tool for comparing images of people. In Ť, Proceedings of the 16th IEEE Conf. on Vision Interface, 2003.12.

[6] A.Perera F. Wheeler X. Liu J. Rittscher T. Sebastin T. Yu P. Tun G. Doretto, N. Krahnstoever and K. Harding. “An intelligent video framework for homeland protection”, In Proceedings of SPIE Defense and Security Symposium -Unattended Ground, Sea, and Air Sensor Technologies and Applications IX, 2007.

[7] I. Kitahara K. Kogure U. Park, A. Jain and N. Hagita. Vise: Visual search engine using multiple networked cameras. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR’06)-Volume 03, 2006.

[8] T. Tan L. Wang W. Hu and S. Maybank. A survey on visual surveillance of object motion and behaviors. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSŮPART C: APPLICATIONS AND REVIEWS, 2004.

[9] D. Reid, “An algorithm for tracking multiple targets,” IEEE Trans. Automatic Control, vol. AC-24, pp.843–854, 1979.

[10] I. J. Cox and S. Hingorani. “An efficient implementation and evaluation of Reid’s multiple hypothesis tracking algorithm for visual tracking”. In IAPR-94, 1994.

[11] Shan, Y., Sawhney, H., Kumar, R.: “Vehicle Identification between Non-Overlapping Cameras without Direct Feature Matching”. Computer Vision. IEEE International Conference on 1 (2005).

[12] Guo, Y., Shan, Y., Sawhney, H., Kumar, R.: PEET: Prototype Embedding and Embedding Transition for Matching Vehicles over Disparate Viewpoints. Computer Vision and Pattern Recognition. IEEE Computer Society Conference on (2007).

[13] Gheissari, N., Sebastian, T., Hartley, R.: Person Re-identification Using Spatiotemporal Appearance. Computer Vision and Pattern Recognition. IEEE Computer Society Conference on (2006) 1528–1535.

[14] Comaniciu, D., Ramesh, V., Meer, P.: "Real-time tracking of non-rigid objects using mean shift. Computer Vision and Pattern Recognition". IEEE Computer Society Conference on 2 (2000).

[15] Varma, M., Zisserman, A.: "A Statistical Approach to Texture Classification from Single Images". International Journal of Computer Vision 62(1) (2005) 61–81.

[16] Dalai, N., Triggs, B., Rhone-Alps, I., Montbonnot, F.: "Histograms of oriented gradients for human detection". Computer Vision and Pattern Recognition. IEEE Computer Society Conference on 1 (2005).

[17] Huang, J., Ravi Kumar, S., Mitra, M., Zhu, W., Zabih, R.: Spatial Color Indexing and applications, International Journal of Computer Vision 35(3), 245–268 (1999).

[18] Birchfield, S., Rangarajan, S.: Spatiograms versus Histograms for Region-Based Tracking. Computer Vision and Pattern Recognition. IEEE Computer Society Conference on 2 (2005).

[19] Hadjidemetriou, E., Grossberg, M., Nayar, S.: Spatial information in multiresolution histograms. Computer Vision and Pattern Recognition. IEEE Computer Society Conference on 1 (2001) 702–709.

[20] Javed, O., Shafique, K., Shah, M.: Appearance Modeling for Tracking in Multiple Non-overlapping Cameras. Computer Vision and Pattern Recognition. IEEE Computer Society Conference on 2 (2005) 26-33.

[21] Gray, D., Brennan, S., Tao, H.: Evaluating Appearance Models for Recognition, Reacquisition, and Tracking. Performance Evaluation of Tracking and Surveillance (PETS). IEEE International Workshop on (2007).

[22] EC Funded CAVIAR project/IST 2001 37540, found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[23] D. Baltieri, R. Vezzani, R. Cucchiara, "3DPes: 3D People Dataset for Surveillance and Forensics" in Proceedings of the 1st International ACM Workshop on Multimedia access to 3D Human Objects, Scottsdale, Arizona, USA, pp. 59-64, Nov 28 - Dec 1, 2011.

[24] Madden, C., Cheng, E. D., Piccardi, M.: Tracking people across disjoint camera views by an illumination-tolerant appearance representation In: Mach. Vision Appl., vol. 18, n. 3, pp. 233–247 (2007).

[25] Teixeira, L. F., Corte-Real, L.: Video object matching across multiple independent views using local descriptors and adaptive learning, In: Pattern Recognition. Lett.,vol.30,n.2,pp.157-167(2009)

[26] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1993.

[27] C. L. Novak and S. A. Sharer, Anatomy of a Color Histogram, in Proceeding of Computer Vision and Pattern Recognition, Champaign, IL, June, 1992, pp 599-605.

[28] D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91–110, 2004.

[29] Blei, D. M.; Ng, A. Y.; Jordan, M. I., Latent dirichlet allocation, In Journal of Machine Learning Res., MIT, vol.3, pp. 993–1022, (2003).

[30] Russell, B. C., Freeman, W. T., Efros, A. A., Sivic, J., Zisserman, A.; Using Multiple Segmentations to Discover Objects and their Extent in Image Collections; In IEEE Proc. of CVPR 2006, vol. 2, pp. 1605–1614 (2006).

[31] Richard O. Duda, Peter E. Hart, David G. Stork; “Pattern Classification”, Wesley, 2000.

[32] Andrew Moore, tutorial slides, http://www.autonlab.org/tutorials/.

[33] R. Schapire. The boosting approach to machine learning: An overview. In MSRI Workshop on Nonlinear Estimation and Classification, 2001.

[34] Fei-Fei L. and Perona P., "A Bayesian Hierarchical Model for Learning Natural Scene Categories", Proc. of IEEE Computer Vision and Pattern Recognition. (CVPR), pp. 524-531, (2005).

[35] Kadir T., Brady M., "Saliency, Scale and Image Description", Int. J.Comput. Vision, vol. 45, pp. 83 - 105, 2001.