CVLab: Computer Vision Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
http://cvlab.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Multi-camera face detection and recognition applied to people tracking

Master Thesis

Michalis Zervos

Supervisor	**Professor Pascal Fua**
Teaching Assistant	**Horesh Ben Shitrit**

Autumn Semester

January 2013

ABSTRACT

This thesis describes the design and implementation of a framework that can track and identify multiple people in a crowded scene captured by multiple cameras. A people detector is initially employed to estimate the position of individuals. Those positions estimates are used by the face detector to prune the search space of possible face locations and minimize the false positives. A face classifier is employed to assign identities to the trajectories. Apart from recognizing the people in the scene, the face information is exploited by the tracker to minimize identity switches. Only sparse face recognitions are required to generate identity-preserving trajectories. Three face detectors are evaluated based on the project requirements. The face model of a person is described by Local Binary Pattern (histogram) features extracted from a number of patches of the face, captured by different cameras. The face model is shared between cameras meaning that one camera can recognize a face relying on patches captured by a different camera. Three classifiers are tested for the recognition task and an SVM is eventually employed. Due to the properties of the LBP, the recognition is robust to illumination changes and facial expressions. Also the SVM is trained from multiple views of the face of each person making the recognition also robust to pose changes. The system is integrated with two trackers, the state-of-the-art Multi-Commodity Network Flow tracker and a frame-by-frame Kalman tracker. We validate our method on two datasets generated for this purpose. The integration of face information with the people tracker demonstrates excellent performance and significantly improves the tracking results on crowded scenes, while providing the identities of the people in the scene.

SUBJECT AREA: Computer Vision

KEYWORDS: Face Detection, Face Recognition, People Tracking

The completion of this thesis was part of the graduate program that was co-funded through the Act “I.K.Y Bursary Program process with individualized assessment, academic year 2011-2012” by the O.P. resources "Education and Lifelong Learning" of the European Social Fund (ESF) and the NSRF 2007-2013.

CONTENTS

ACRONYMS

1-NN / 3-NN / k-NN	1 Nearest Neighbor / 3 Nearest Neighbors / k Nearest Neighbors
BBF	Binary Brightness Features / Control Point Features
KSP	K-Shortest Paths
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
MCNF	Multi-Commodity Network Flow
MODA	Multiple Object Detection Accuracy
MOTA	Multiple Object Tracking Accuracy
NCC	Normalized Cross Coefficient
PCA	Principal Component Analysis
POM	Probabilistic Occupancy Maps
RBF	Radial Basis Function
ROI	Region Of Interest
SVM	Support Vector Machine
T-MCNF	Tracklet based Multi-Commodity Network Flow
VJ	Viola – Jones
ZM-NCC	Zero-meaned Normalized Cross Coefficient

1. INTRODUCTION

Identifying and tracking people in a crowded scene is a complex problem with many possible applications. Those two problems can be approached separately but when combined into a single framework they can provide better results by sharing information. Using the identity of the people can help the tracker to minimize switches, while using the trajectories produced by the tracker can give information about the identity of a person, when no appearance information is available. In this thesis the facial characteristics are exploited to boost the performance of a tracker and provide identities to the tracked people.

Lately, there has been a lot of work on the domain of tracking-by-detection [1]. Those methods rely on an object (people in our case) detector that generates probabilities of a person being at a (discretized) point on the ground plane of the scene at each frame. Those detections are linked together to form trajectories. When no other information is available and the detections are noisy, the resulting trajectories might be inconsistent and contain identity-switches between the tracked people. By exploiting the facial characteristics we minimize the identity switches and identify the people being tracked.

The goal of this thesis is to design and build a complete system for identification and tracking. This is a two-step process, first there is an offline procedure where the face model of each individual is captured and stored. Then, people in the scene can be reliably tracked and identified. There are six connected components that make this system work: a) a people detector which gives probabilities of people standing at a point in the scene, b) a face detector which searches for faces at the locations were people are expected to be found, c) a face modelling technique to capture the information of each individual which can then be stored to d) a face database and used by the e) face recognition algorithm to identify the individuals. Those recognitions are finally used by the f) people tracker to track their movement in the space and preserve their identities. A diagram that shows the overview of the system is shown in Figure 1.

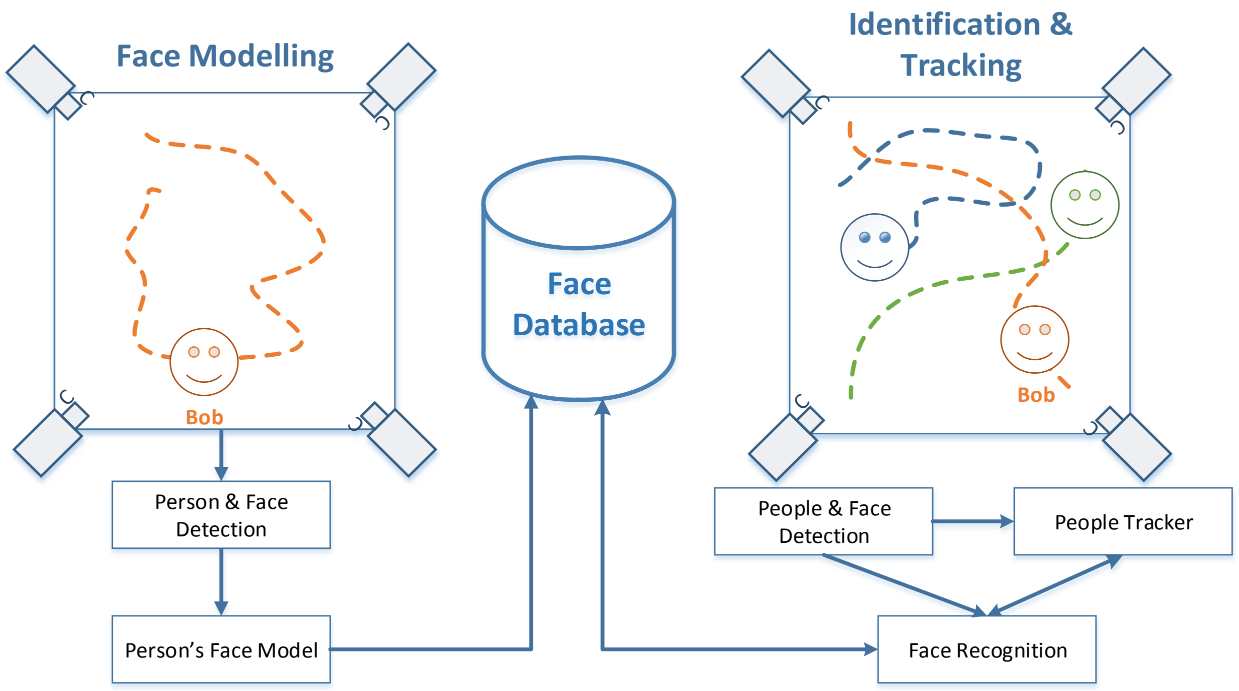


Figure 1 – The complete system overview, containing the different modules and their connection.

Information about the identity of a person needs only be available in a limited number of frames, for the tracker to be able to form consistent identity-preserving trajectories. Those sparse recognitions, though, should be as reliable as possible, so throughout the thesis we target for high precision and low false positive rates. People that are not included in the database can still be tracked but are marked as guests. The system can learn a face model from some cameras and use this information later to identify a person by a different camera, effectively transferring the face model between cameras.

Such a system could be used in many applications. One such scenario would be in a commercial store, where the database could contain information about the employees and the application would track the people in the store. The customers (guests) could be distinguished by the employees and this can give meaningful information in many ways, like: detecting where people spend more time and analyzing their trajectories so that products or advertisements would be placed in more visible locations, analyzing the interaction between customers and employees, optimizing the distribution of employees around the shop or it can go even further and detect customers that appear to be looking for an employee to assist them at real-time, etc.

1.1 Thesis Structure

The rest of this thesis is organized as follows. In chapter 2 we present the related work. Chapter 3 is devoted to the face detection methods that were tested and the final form of the face detector we implemented. We continue by analyzing the face recognition method on chapter 4. The integration with two object tracking methods is presented in chapter 5. Some implementation information of the system and the supporting software that was implemented during the thesis can be found in chapter 6, followed by the conclusions in chapter 7.

2. LITERATURE REVIEW

Our detection, identification and tracking framework is composed of several modules, each of which deals with a separate problem. In this chapter we present the research work on each of those fields separately and also discuss similar framework approaches that combine detection, tracking and identification. All those fields are quite mature, with hundreds of different methods proposed over the years, so we limit the discussion to only a few of those and provide references to surveys of each field.

2.1 Face detection

Face detection has been one of the most studied topics in the computer vision literature. The goal of face detection is given an arbitrary image containing an unknown number of faces (possible none) to localize the face and determine its size. This is a challenging task considering the variations in pose, lighting, facial expression, rotation, scale, and occlusions. Hundreds of methods have been proposed in the last years.

Probably the most well-known method that dramatically influenced the research on the field and allowed for out-of-lab applications is the Viola – Jones method [2]. Since this is one of the methods that we evaluated for our system, we extensively present this work in Appendix A. The Viola – Jones method was very successful because of three elements; the computation of the Haar-like features on the integral image, the AdaBoost learning method and the cascade of classifiers it employed. All these ideas were then further extended and lead to many other face detection methods.

A similar approach was proposed in [3], where instead of computing Haar-like features, individual pixels are compared making the method faster. This is the method that we use in this thesis. The original work of Viola and Jones considered only 4 types of horizontal / vertical features. This was later extended by Lienhart and Maydt [4] by introducing rotated features. In the following years many variations of the Haar-like features where proposed. The interested reader is referred to [5] for more information.

Another category of features that is widely used in face detection is based on histograms that capture regional statistics. *Levi* and *Weiss* [6] proposed the use of local edge orientation histograms which are evaluated in sub-regions of the search window. Those features are then combined using a boosting algorithm. Detectors based on more variations of the edge orientation histograms were later proposed, with the histograms of oriented gradients (HoG) being the most popular of those.

The most widely learning technique used in the face detection field is the AdaBoost or some variation of that method, combined with a cascade classifier. However, different types of learning methods have been proposed, some of which performed equally well. Neural networks [7], SVMs [8] and other machine learning schemes have been proposed for solving the face detection problem.

In multi-view face detection, usually, a collection of detectors is trained, one for each family of poses and during detection a pose estimator is employed to decide which classifier should be queried [8]. A different approach is proposed in [9], where the authors train a single classifier that has the ability to deform based on the image. The method consists of a set of pose estimators that can compute the orientation in parts of the image plane and allow a boosting-based learning process to choose the best combination of pose estimators and pose-indexed features. The detector can adapt to appearance changes and deformations, so there is no need to fragment the train data into multiple sets, one for each different pose family.

A very different approach that combines the problem of tracking and detecting a face in a video sequence was proposed by *Kalal et al.* in [10], which is an extension of his original TLD (Tracking – Learning – Detection) approach [11]. A generic face detector and a validator are added to the original TLD design. The off-line trained detector localizes frontal faces and the online trained validator decides which faces correspond to the tracked person.

Recently, *Zhu* and *Ramanan* [12] proposed a unified model for multi-view face detection, pose estimation and landmark (eyes, mouth, nose, chins) localization that advanced the state-of-the-art results in multiple standard benchmarks and in a new “in the wild” pictures dataset. Their model is based on a mixture of trees with a shared pool of parts; each facial landmark is modelled as a part and global mixtures are used to capture topological changes due to viewpoint.

For an extensive survey of the face detection topic the reader is referred to [5], [13].

2.2 Face recognition

The goal of face recognition is, given a face image and a database of known faces, to determine the identity of the person in the query image. Like the face detection problem, it has also been studied a lot the past few years. Face recognition also shares some of the difficulties that arise in the face detection problem, like variations in lighting, facial expression or pose.

Probably the most well-known method for face classification is the work of *Turk* and *Pentland* [14], which is based on the notion of eigen-faces. It was earlier shown [15] that any face image can be represented as a linear combination of a small number of pictures (called eigenfaces) and their coefficients. The method is based on the Principal Component Analysis (PCA); the eigenfaces are the principal components of the initial training set of images. Each image (size of N pixels), is considered as an N-vector and the eigenfaces correspond to the eigenvectors of the covariance matrix. The eigenvectors can be efficiently extracted using the Singular Value Decomposition (SVD). So each face image can be represented as a linear combination of those eigenfaces, but it can be very well approximated using just a few of the first eigenvectors (those with the largest eigenvalues). Those K eigenvectors (eigenfaces) create a K-dimensional space, the face space. An unknown face image can be classified by projecting it to the face space; this results in a vector containing the coefficients of the new image. By simply taking the Euclidean distance between this vector and the ones of each known class/face, one can find the class which is closest to the query face.

Even though the PCA yields excellent results when it comes to representing a face with a few coefficients, when the goal is classification there is room for improvement. The face space generated by the PCA is one where the variance of all the samples is minimized. By projecting the faces in a space where the variance of the samples within one class is minimized and on the same time the variance of the samples between classes is maximized, one can get better classification results. This is the idea behind the Fisher-faces method [16]. The basis vectors for such a sub-space can be generated by Linear Discriminant Analysis (LDA). A more detailed comparison of the Eigenfaces (PCA) versus the Fisherfaces (LDA) methods can be found in [17] and [18].

The above two methods are holistic, in the sense that use the global representation of the face. These methods have the advantage that the exploit all the information of the image without concentrating on small regions of it. On the other hand they are not very robust to illumination changes, misalignment and facial expressions. This is where feature-based methods perform better. A method that combines the benefits of both worlds is the Local Binary Pattern (LBP) Histograms approach [19], which encodes the facial information of a small region into features but also takes into account the spatial configuration of those areas. An 1-NN is employed for the classification. In this thesis we exploit the same features but an SVM is used for classification.

Over the last few years, probably due to the wider availability of depth sensors, there has been considerable research in the area of 3D face recognition. Precise 3D models of the face can be built, providing much more information about the facial characteristics and leading to better recognition rates. A good analysis of recent 3D face recognition techniques is provided in [20].

Most face recognition methods are designed to work best on images (usually of frontal faces). However, there has been some work on face recognition on video sequences. When dealing with videos, more effective representations such as 3D models, discussed above, can be generated and exploited. Also the face model can be updated over time as shown in [21]. More information on video-based face recognition can be found in [22], a recent survey of the topic.

Recently, research work has shifted towards recognition on uncontrolled environments where illumination changes, blurring and other degradation is observed. In [23], the authors propose a novel blur-robust face image descriptor based on Local Phase Quantization and extend it to a multi-scale framework (MLPQ). The MLPQ method is combined with Multi-Scale Local Binary Patterns (MLBP) using kernel fusion to make it more robust to illumination variations. Finally, Kernel Discriminant Analysis of the combined features can be used for face recognition purposes. The method produces state-of-the-art results in various datasets where illumination changes, blur degradation, occlusions and various facial expressions are encountered.

The reader is referred to the survey [24] for more information on the topic.

2.3 People tracking

Tracking multiple objects (people in our case) in a scene is a well-studied area of computer vision. We focus this review on methods that are similar to the proposed framework. A very widely used technique is the Kalman filtering. *Black* and *Ellis* [25] propose a multi-camera setup, similar to the one used in this thesis, to resolve object occlusion. A Kalman filter is used to track objects in 3D in the combined field of view of all cameras. *Mittal* and *Davis* [26] also rely on Kalman filter to track multiple people seen by many cameras. In this case, however, the tracking is done on the 2D ground plane. The location of the people is estimated by using evidence collected from pairs of cameras. The main drawback of all the approaches based on the Kalman filter is the frame-by-frame update which quite often leads to identity switches in crowded scenes.

Particle filter-based methods only partially overcome this problem while they tend to be computationally intensive. In [27] an unknown number of interacting targets is tracked by a particle filter that includes a Markov Random Field motion prior that helps preserving identities throughout interaction. The sampling step of the particle filter is replaced by Markov Chain Monte Carlo sampling which is more computationally efficient for a large number of targets. *Okuma et al.* [28] propose a method that combines mixture particle filters and a cascaded AdaBoost detector to track hockey players.

Another approach to multi-target tracking is to generate high-confidence partial track segments, called tracklets, and then associate them into trajectories. The optimal assignment in terms of some cost function can be computed using the Hungarian algorithm. *Singh et al.* [29] utilize a robust pedestrian detector to generate unambiguous tracklets and then associate the tracklets using a global optimization method. Unlike other tracklet-based methods, they also exploit the unassociated low confidence detections between the tracklets to boost the performance of the tracker. *Henriques et al.* [30] propose a graph-based method to solve the problem of tracking objects that are merged into a single detection. When the people are merged together, their identities can no longer be used to track the reliably. They formulate this as a flow circulation problem and propose a method that solves it efficiently and tracks the group as a single identity. The final trajectories are generated by merging and splitting tracklets using a global optimization framework.

Recently, the multi-target tracking has been formulated as a Linear Programming (LP) problem that leads to a globally optimal solution across all frames. Given the noisy detections in individual frames, one can link detections across frames to come up with trajectories. When dealing with many objects, the linking step results in difficult optimization problem in the space of all possible families of trajectories. The KSP method [31] formulates the linking step as a constrained flow optimization problem that leads to a convex problem. Due to the problem's structure, the solution can be obtained by running the k-shortest paths algorithm on a generated graph. An extension to the KSP method, is the Multi-Commodity Network Flow (MCNF) tracker [32] which takes into account appearance information to minimize the identity switches and provide state-of-the-art results. This is the tracker we employ and combine with our face recognition method.

2.4 People tracking with identities

Mandeljc et al. [33] proposed a novel framework that fuses a radio-based localization system with a multi-view computer vision approach, to localize, identify and track people. A network of Ultra-Wideband sensors in the room and radio-emitting tags worn by people provides identity information which is combined with a computer vision method for localizing individuals. By fusing the information, the authors report excellent performance that outperforms the individual parts. The use of the radio network addresses the common problem of identity switching appearing in computer vision methods, by providing reliable identity information. The KSP method [31], discussed above, combined with the POM people detector [34] is employed as the computer vision framework. The combined approach resembles the idea of this work where the MCNF tracker, exploits identity cues (face information) to identify people and generate identity-preserving trajectories.

Recently, *Cohen* and *Pavlovic* [35] proposed a unified framework for multi-object tracking and recognition where the two problems were solved simultaneously. The joint problem was formulated as an Integer Program (IP) optimized under a set of natural constraints. Since solving the IP problem is computationally inefficient, the problem is reformulated as a Generalized Assignment Problem (GAP), instead of relaxing the problem, as done by other methods. An efficient approximate combinatorial algorithm is used to solve the GAP which guarantees the approximation bounds. Given the set of faces in the database and the set of unidentified detections in the scene, the goal is to find an assignment between the two sets that minimizes the total cost, which is the sum of the cost of each assigned pair. The cost for each mapping between a detection and a face in the database is given by an SVM face classifier. The faces are described by Local Binary Pattern (LBP) features. A number of constraints limit the possible solutions. Two types of constraints are introduced; temporal constraints that essentially penalize inconsistencies between consequent frames and pairing constraints that express that a known face can only appear at most once in each frame and that one detection can be paired with at most one known face.

3. FACE DETECTION

In this chapter we present the various face detection methods that were analyzed and evaluated in order to choose the one that suits our goals best. Obviously, we target for the best possible accuracy but the computational efficiency is of great importance. On the last section of this chapter (3.5) we analyze the method that is eventually used on our framework.

3.1 Viola - Jones Method

The Viola – Jones [2] method is always a contender for robust face detection. The method computes set of Haar-like features at multiple scales and locations and uses them to classify an image patch as a face or not. A simple, yet efficient, classifier is built by choosing a few effective features out of the complete set of the Haar-like features that can be generated using the AdaBoost [36] method. A number of classifiers, ranging from a very simple 2-features one up to more complex layers containing many features, are combined in a cascade structure to provide both accuracy and real-time processing. More information about the method can be found at Appendix A.

3.2 Control Point Features (Binary Brightness Features – BBF) Method

Abramson et al. [3] propose an improvement to the detection method of the initial Viola and Jones framework. Instead of focusing on rectangular areas for their features, they examined individual pixel values.

The advantage of this method lies in the simplicity of the proposed features, meaning that the detection process is much faster than the Viola and Jones method. To compute the control point features, there is no need:

a) to compute any sum of a rectangle, thus an integral image isn't prepared at all and
b) to normalize the variance (and the mean) of the search window.

3.2.1 Binary features

Rather than computing the difference of the sums of rectangular areas and having a threshold for each feature, the control point features are "binary" and compare the intensity values of individual pixels (called control points) outputting 1 or 0 ("greater" or "less or equal").

A feature consists of two sets of control points $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_m\}$ where $n, m \leq K$. The constant K regulates the maximum number of control points per set. The trade-off between speed performance and accuracy can be controlled by changing this value. The authors experimented with different values and set $K = 6$ because it demonstrated good speed, while higher values didn't improve the results substantially. Given an image windows of size $W \times H$ (24 x 24 pixels is used in the paper experiments), each feature is evaluated either one of the following three resolutions:

1. The original $W \times H$
2. Half – size $\frac{1}{2}W \times \frac{1}{2}H$
3. Quarter – size $\frac{1}{4}W \times \frac{1}{4}H$

So each feature f is described by the two sets of control points X_f, Y_f and the resolution on which it works on. The feature f is evaluated to 1 if and only if all the control points of the set X_f have values greater than all of those in Y_f:

$$f = \begin{cases} 1, & \forall x \in X_f, \forall y \in Y_f : I(x) > I(y) \\ 0, & otherwise \end{cases} \quad (1)$$

Where $I(p)$ denotes the intensity value of the image sub-window at point p.

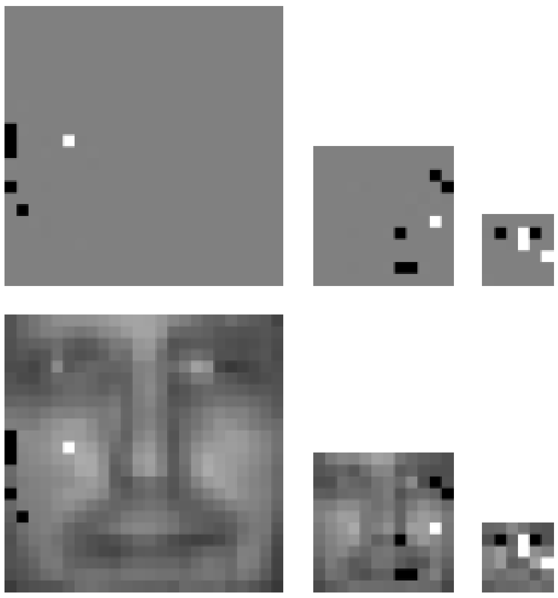


Figure 2 - Sample of 3 control point features, one on each of the three resolutions. Bottom row shows the pixel positions overlayed on the image patch (image from [3]).

Obviously no variance normalization is required since the algorithm only checks the sign of the difference of two pixels, and not the actual difference. Three examples, one for each resolution, of the Binary Brightness Features (or Control Point Features) can be seen in Figure 2. The X set is represented by the white pixels, while the Y set is represented by the black pixels.

3.2.2 Training

The BBF method, like the Viola – Jones one, relies on AdaBoost [36] for the training of the final classifier. As in the Viola – Jones method each weak classifier corresponds to one feature. However, to select the feature for each round of the AdaBoost a genetic algorithm is employed. It would be impossible to test all the possible features, since their number for a 24 x 24 pixels window is about 10^{32}. The genetic algorithm begins with a generation of 100 random simple classifiers (feature). At each iteration, a generation of 100 classifiers is produced by the previous one as follows:

1. The 10 classifiers that produced the lowest error are retained, while the other 90 are discarded.
2. 50 new features are generated by mutations of the top 10 that was kept on step 1. Possible mutations are:
    a. Removing a control point
    b. Adding a random control point
    c. Randomly moving an existing control point within a radius of 5 pixels
3. Another 40 random features are generated

For each generation the single best feature is kept if the error has improved with respect to the best feature of the previous generation. The algorithm stops if there is no improvement for 40 generations and the best feature is selected for the iteration of the boosting algorithm. A strong classifier is built in the end of the boosting process.

Just like the Viola – Jones framework, the final classifier is a cascade of strong classifiers created as described above. The first layer contains the 2 features shown in Figure 3 (a) at half resolution, while the second layer contains the 3 features shown in Figure 3 (b) in half, quarter and original resolution respectively.

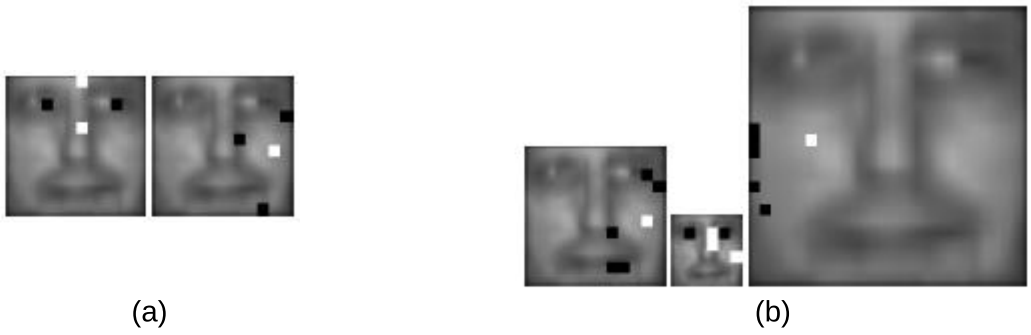


Figure 3 - The features of the first two layers of the cascade classifier overlayed on image patches (from [3])

The first feature of the first layer and the second feature of the second layer resemble the second feature of the Viola – Jones method, which captures the fact that the area between the eyes should be brighter than the area of the eyes. The second feature of the first layer and the third feature of the second layer encode the information that the chick is usually brighter than the surrounding edges of the face.

3.2.3 Detecting faces in an image

Instead of one cascaded classifier, actually three were trained. One using the original 24x24 pixels window, one having 32x32 pixels window and one with window size 40x40. To detect faces in new images, the classical sliding window approach is employed. A simple image pyramid is built where each level is half the size of the previous one. For each level of the pyramid, the 24x24 window slides along the image and the features are checked. To check for the half-size and quarter-size features, the two pyramid levels below are used. The same procedure is repeated for the 32x32 and 40x40 detectors. The union of all the detections of the 3 detectors in all levels of the pyramids are the resulting detections. Similar detections are united into one.

3.3 Deformable detector

We also evaluated the deformable detector from [9] which provides state-of-the-art performance. The proposed method is based on a classifier that can adaptively deform to detect the target object. It relies on a set of pose-indexed features [37] augmented with a family of pose estimators. In contrast with other similar object-detection methods based on pose-indexed features, it doesn't require labelling for rigid rotations and deformations. A family of pose estimators provides estimates of rigid rotations and deformations from various areas of the image. A learning method based on AdaBoost chooses the best combination of pose-indexed features and pose estimators. This means that a pose-indexed feature may obtain the pose estimate form a different area of the image than the one that the response is computed on. Finally, a flexible detector is generated by weighted features, each optimized with the best pose estimate.

Three types of features are combined in the final detector: a) Standard features that check for the absence/presence of edges at a specific location and orientation, b) Pose-indexed features that have a fixed location but check for edges in an orientation depending on the dominant orientation as given by a pose-estimator for an area of the image and c) Pose-indexed features whose location and orientation extraction depend on the dominant orientation given by a pose estimator.

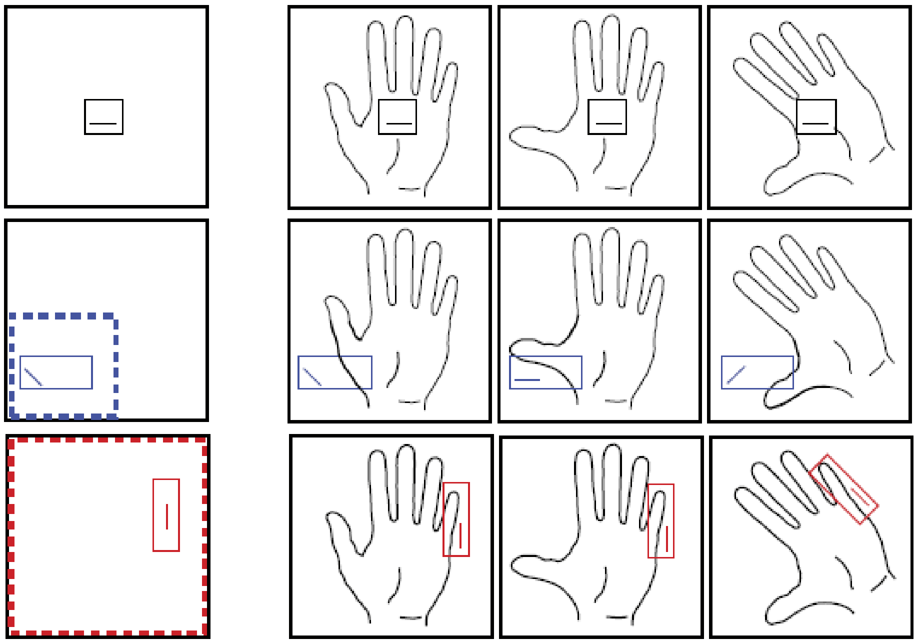


Figure 4 – The three different types of features (one on each of the three rightmost columns) used in the deformable detector of [9]. Left column shows the feature position (solid box), the dominant orientation (line) and the pose estimator support area (dashed box). Each row contains a different instance/examples of a hand.

The three types of features are demonstrated in Figure 4 for three different poses of a hand (3 right columns). First row shows a standard feature, the second row shows a pose-indexed feature whose location is fixed but the orientation depends on the pose-estimator and the third row shows a pose-indexed features whose location and orientation depends on the pose-estimator. The example features are shown in the first column. The solid line box shows the area where the edge-counting feature is calculated, the solid line in the box shows the orientation of the extracted feature, the dashed color box shows the area from where the pose-estimate is computed.

The running time of the detector was prohibitively large for the requirements of this project (image resolution, real-time performance) so this option was dropped. Since the framework is quite modularized, this method (or any other face detection method) can be used in the future with minor changes in the code.

3.4 Comparison of the methods

In this section we present the result of the experiments we conducted with the above mentioned algorithms. The methods are evaluated on two publicly available datasets; one image database, the MIT+CMU frontal faces[1] and a video sequence, the Terrace dataset[2]. First we define the metric that was used (MODA) to compare the algorithms and then we present the comparison results.

[1] MIT+CMU Face Dataset: http://vasc.ri.cmu.edu//idb/html/face/frontal_images/index.html
[2] Terrace Videos Sequence: http://cvlab.epfl.ch/data/pom/#terrace

3.4.1 Multiple Object Detection Accuracy (MODA)

The evaluation of the face detectors was based on the NMODA (N - Multiple Objects Detection Accuracy) score as defined in [38]. The MODA metric for a single frame / image t is calculated by the following formula:

$$MODA(t) = 1 - \frac{C_m(m_t) + c_f(fp_t)}{N_G^{(t)}} \quad (2)$$

Where m_t, fp_t are the number of missed detections and false positives respectively, c_m, c_f are cost functions (equal to 1 for our experiments) and N_G are the number of ground truth objects.

$$If\ N_G^{(t)} = 0 \begin{cases} MODA(t) = 0, & if\ C_m(m_t) + c_f(fp_t) > 0 \\ MODA(t) = 1, & if\ C_m(m_t) + c_f(fp_t) = 0 \end{cases} \quad (3)$$

For a video sequence of N frames or multiple images, the N-MODA is defined as:

$$NMODA = 1 - \frac{\sum_{i=1}^{N} C_m(m_i) + c_f(fp_i)}{\sum_{i=1}^{N} N_G^{(i)}} \quad (4)$$

A detection is considered a hit if the overlap ratio $\mathrm{r} \in [0,1]$ between the detection D_i and the corresponding ground truth object G_i is above a threshold θ. The overlap ratio is defined as:

$$overlap\_ratio(G_i, D_i) = \frac{G_i \cap D_i}{G_i \cup D_i} \quad (5)$$

To find the correspondences between the detections and the ground truth objects of a frame, the overlap ratio between every pair of detection and ground truth object is calculated and then the Hungarian algorithm (Appendix B) is employed to find the best matches. Figure 5 shows a frame with 3 ground truth objects (blue) and 3 detections (red) and the class (missed, hit, false positive) each of those falls into. On the bottom right case, the overlap ratio is less than the threshold, so it's counted as one false positive and one missed detection.

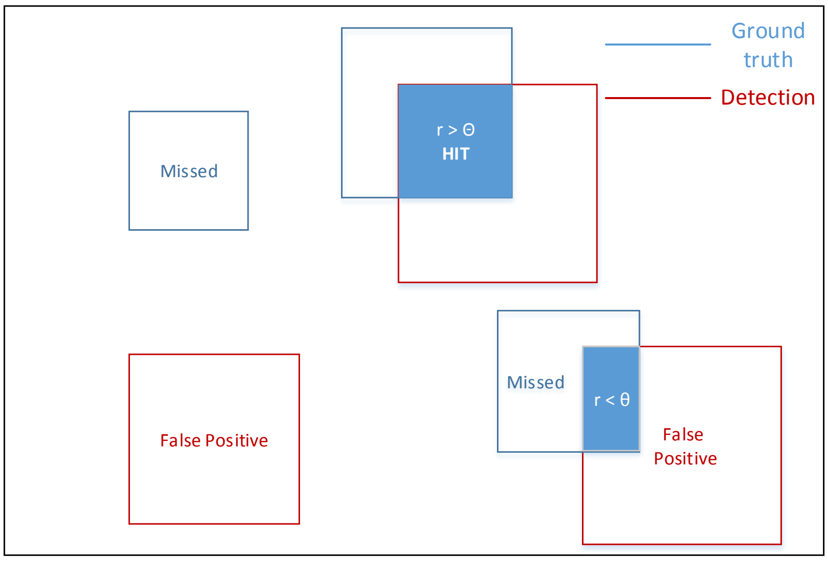


Figure 5 – The 4 different cases of detection - Ground truth object configuration. The bottom right one is counted both as missed detection and false positive.

3.4.2 Images dataset

We evaluated the algorithms on the MIT+CMU frontal faces publicly available dataset [39] that comes with annotations for the landmarks of a face (mouth, eyes and nose). From those landmarks we generated the bounding boxes of the faces and evaluated the algorithms using the MODA score. The dataset consists of 130 images, containing 498 faces in total. There are on average 3.83 faces / image. The average size of the images is 440 x 430 pixels. One of the images in the dataset is shown in Figure 6. The results are presented in Figure 7, while the running times of the algorithms are shown in Figure 8.

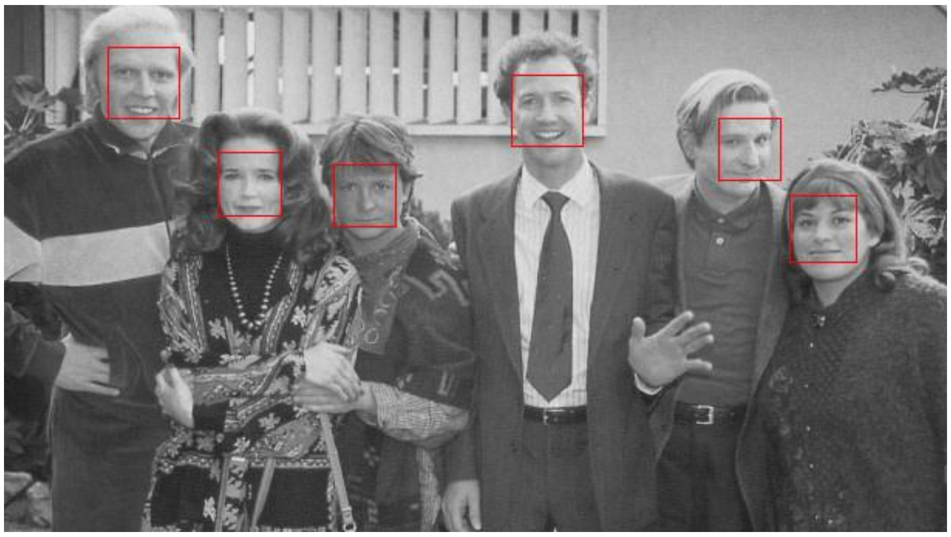


Figure 6 - One of the images of the MIT+CMU dataset with faces detected by BBF method.

On the figures below, *vj* refers to the Viola – Jones and *bbf* to the Binary Brightness Features method. The "*opt*" flag denotes optimized versions of the algorithms that search on fewer space scales to improve the speed.

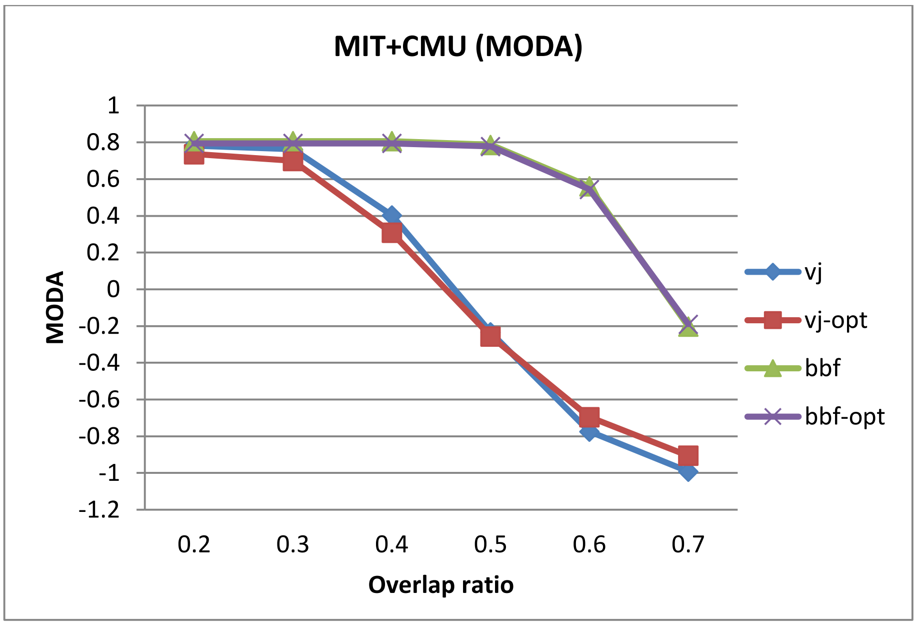


Figure 7 - Face detection evaluation on MIT+CMU dataset. The "vj" refers to the Viola – Jones method and "bbf" to the binary brightness features method. The "opt" are speed optimized versions of the each method. Obviously the BBF method outperforms the Viola – Jones one without any degradation in performance when it's optimized.

As can be seen in the results above the BBF method outperforms the Viola-Jones detector, while the optimized version doesn't show any decrease in performance. It has to be noted that the Viola-Jones method was trained on images of the full face, while the BBF method was trained only on the rectangle starting halfway between the eyes and the top of the head and stopping just under the mouth, and therefore produced detections of that size. The annotations produced by the landmarks, generated rectangles whose size was closer to the ones that BBF produced, that's why the performance of the Viola-Jones method drops significantly as the overlap ratio increases. In any case the BBF method performed at least as good as the Viola-Jones one.

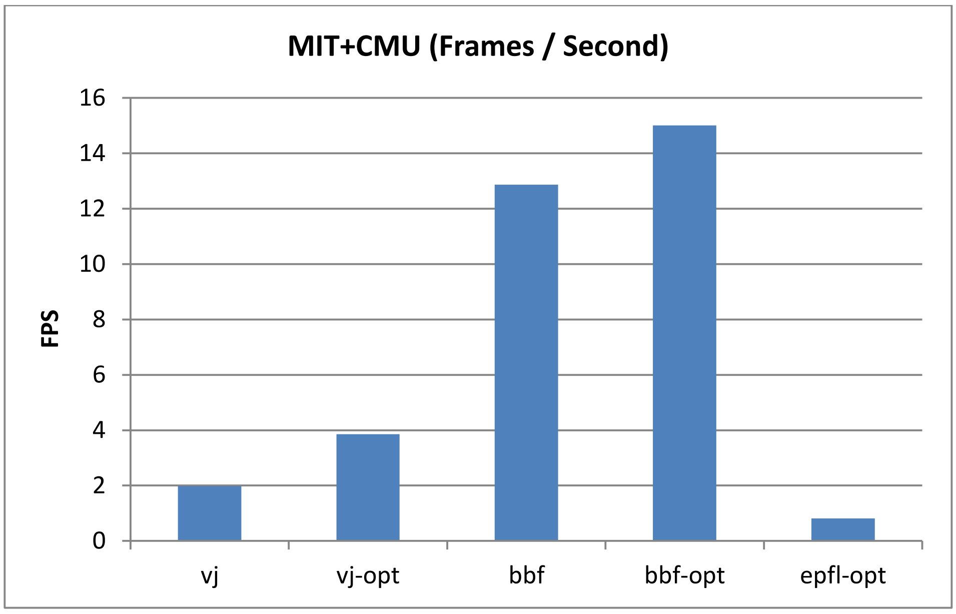


Figure 8 - Frames per second for each algorithm on the MIT+CMU dataset. The "vj" refers to the Viola – Jones method and "bbf" to the binary brightness features method. The "opt" are speed optimized versions of the each method. The BBF method is obviously faster than the rest and the optimized version can reach 15 FPS.

The BBF method appears to be faster than the rest, with the optimized version reaching 15 frames / second. It is 3 – 4 times faster than the Viola – Jones method, which makes sense taking into account the simpler features that are computed on the BBF method. The time to load the image from the file is included in the speed computation of the methods.

3.4.3 Video dataset

We also evaluated the aforementioned algorithms on the 4 videos of the second sequence of the publicly available Terrace dataset. Each video is 3.5 minutes long, shot at 25 fps, totaling to 5250 frames per video encoded using Indeo 5 codec. The size of the frames is 720 x 576 pixels. The detection was performed only at the top half of each frame. A sample frame from this sequence, together with the detections generated by the BBF method is show in Figure 9.The average of the MODA score of the 4 videos is shown in Figure 10 and the running speeds in Figure 11. The faces in the videos were annotated using an application built to that end.

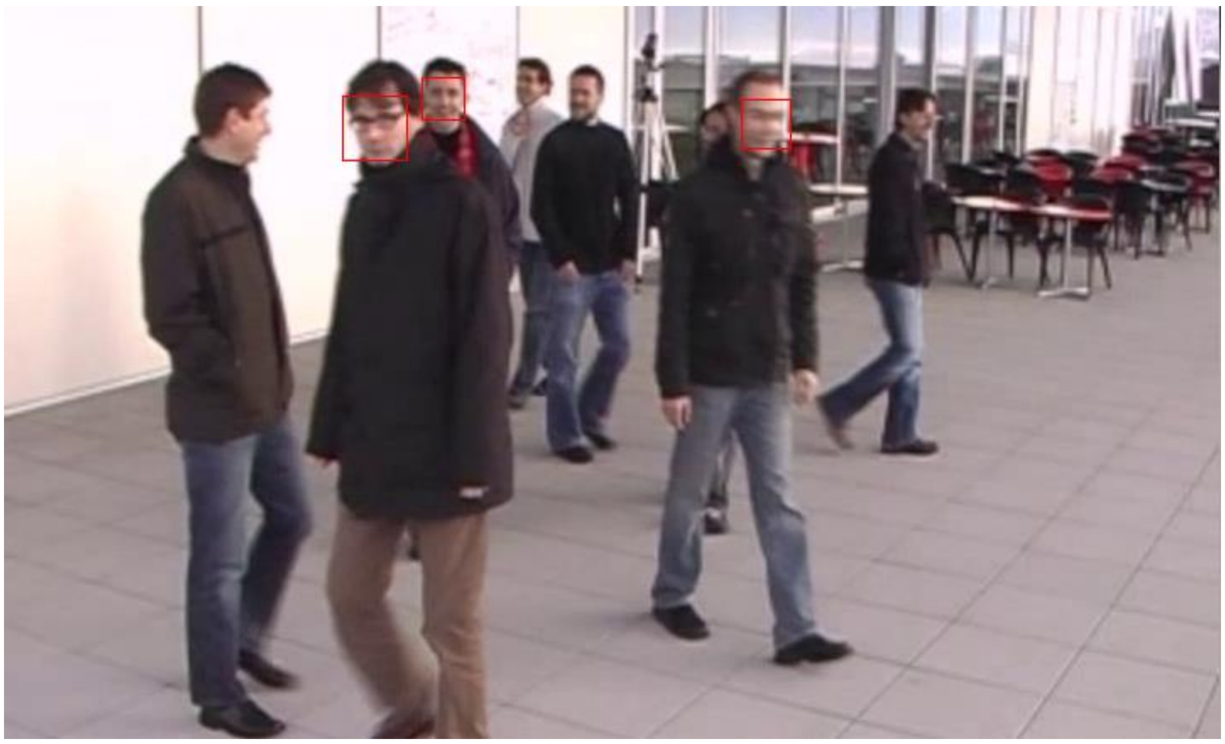


Figure 9 - A frame of the Terrace video sequence with the face detections generated by the BBF method

As before the *bbf-opt* method outperforms the rest, even though the MODA scores are generally too low. This is caused by the very bad video quality (interlacing artifacts) and the video resolution (720 x 576 pixels). However, even in those conditions the *bbf-opt* method appears to be better than the rest. As can be seen below, it's also faster than the others averaging 34 frames per second for a single camera.

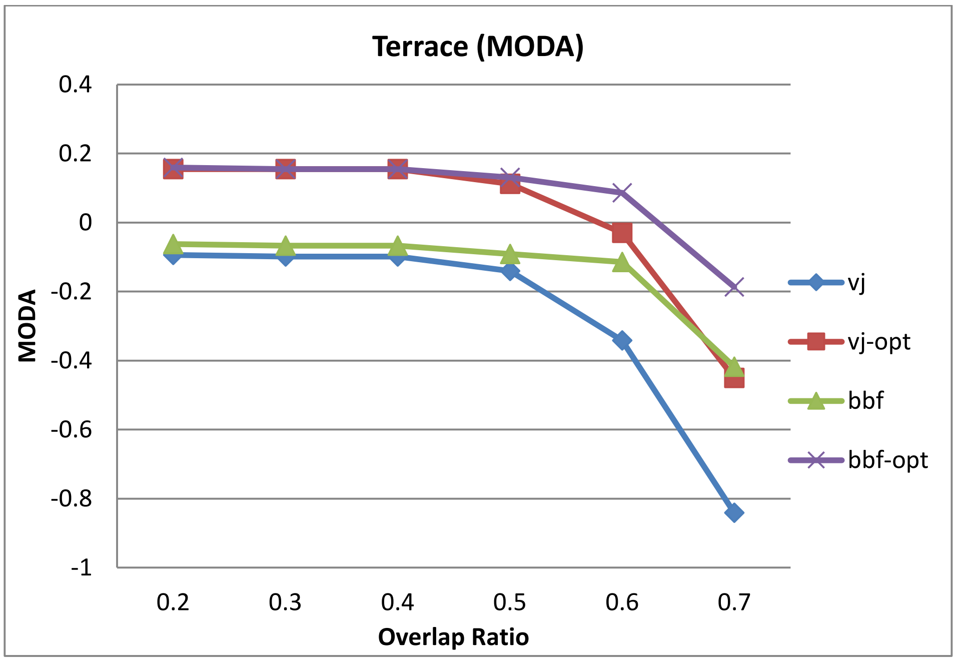


Figure 10 - Face detection evaluation on the Terrace video sequence. The "vj" refers to the Viola – Jones method and "bbf" to the binary brightness features method. The "opt" are speed optimized versions of the each method. Obviously the BBF method outperforms the Viola – Jones one without any serious degradation in performance when it's optimized.

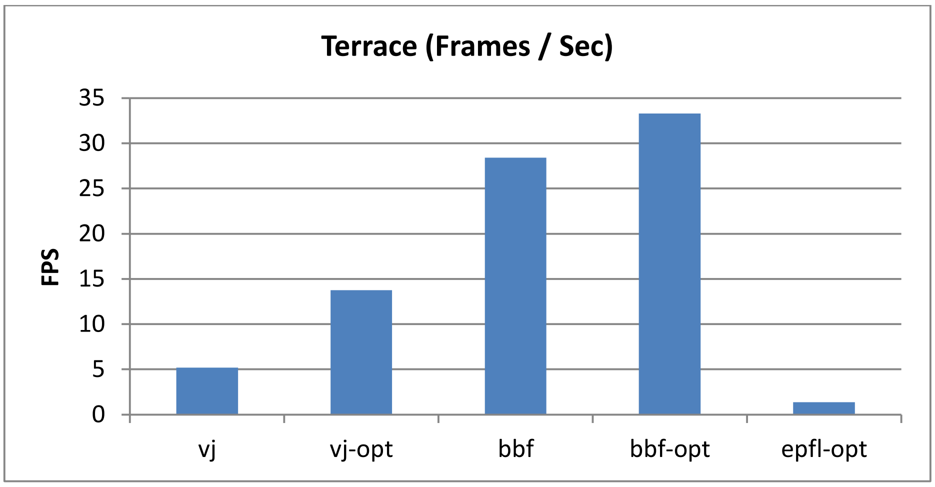


Figure 11 - Frames per second for each algorithm on the Terrace sequence. The "vj" refers to the Viola – Jones method and "bbf" to the binary brightness features method. The "opt" are speed optimized versions of the each method. The BBF method is obviously faster than the rest and the optimized version can reach 34 FPS.

3.4.4 Discussion

Based on the above and the observations on other video sequences that no annotations are available (to provide results in this thesis), we selected the *bbf-opt* algorithm as the face detection method that is further integrated to our system. The low MODA scores on the Terrace video sequence is a result of the bad video quality leading to a large number of false positives. On sequences recorded on the CVLab demo room, using high-end cameras the face detections are consistent; still, some false positives can be noticed.

Figure 12 and Figure 13 demonstrate two cases where false detections appear. To minimize the number of false positives and to achieve much faster detections (since we want to be able to process the input from multiple cameras real-time) we integrate the face detector with a people detector as explained in the following section.

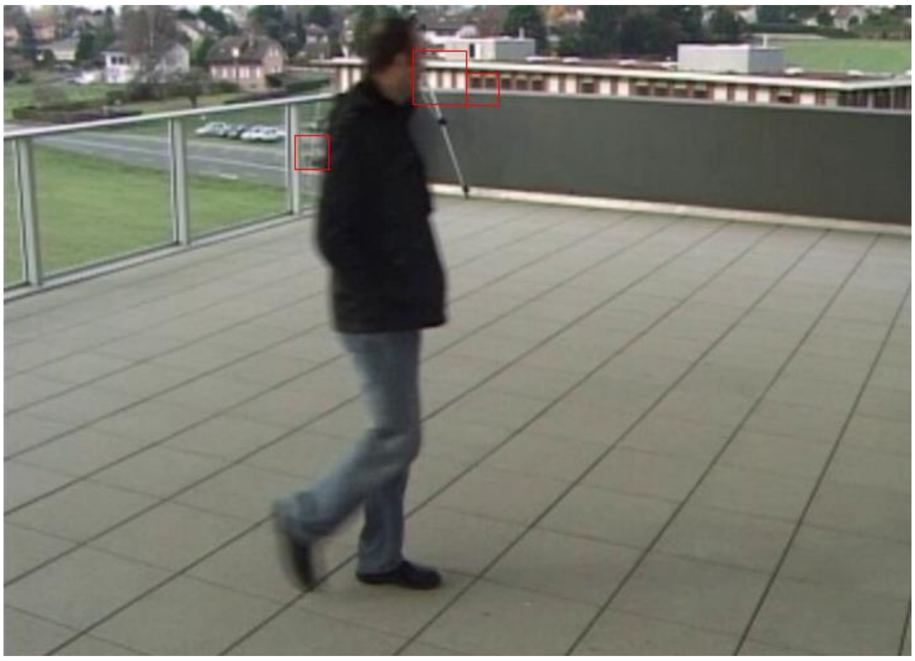


Figure 12 – Example of three false positives detections generated by the BBF method.

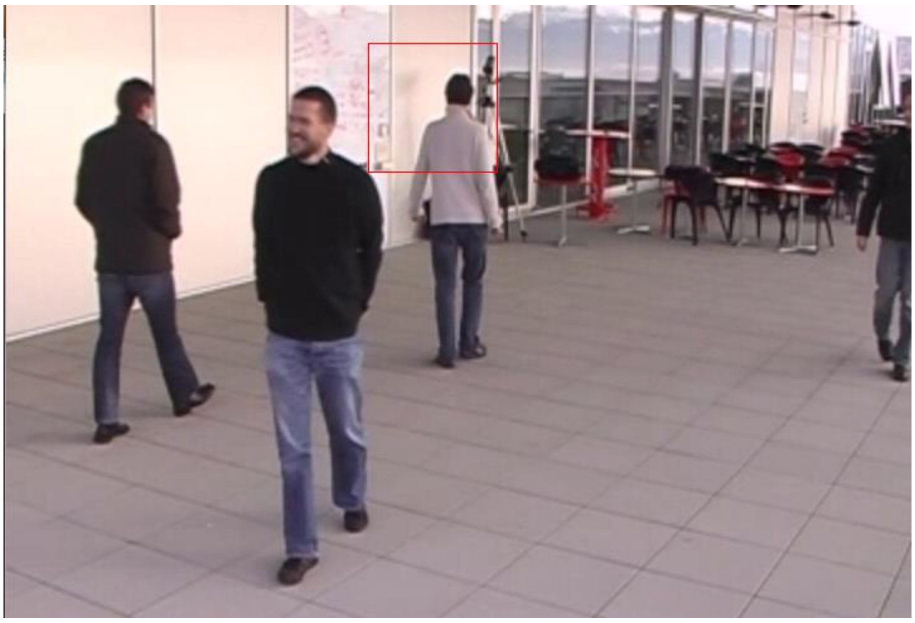


Figure 13 – Example of a false positives detection generated by the BBF method.

3.5 Integration with people detector

In order to improve the false positive rates of the detections, we integrate the BBF face detector, presented above, with the POM (Probabilistic Occupancy Map) people detector [34].

3.5.1 Head location estimation

Le K be the camera calibration matrix of the CCD camera

$$K = \begin{bmatrix} a_x & s & x_0 \\ & a_y & y_0 \\ & & 1 \end{bmatrix} \qquad (6)$$

where a_x, a_y represent the focal length of the camera in terms of pixel dimensions in each direction respectively, $\widetilde{x_0} = (x_0, y_0)$ is the principal point in terms of pixel dimensions and s being the skew. Let P be the 3×4 projection matrix for the camera:

$$P = K[R|t] \qquad (7)$$

where R, t the external parameters of the camera, R being the 3×3 rotation matrix and t the 3×1 translation vector with $t = -R\tilde{C}$ ($\tilde{C}$ being the inhomogeneous coordinates of the camera in center in the world coordinate frame).

We know that a point $\boldsymbol{X} = (X, Y, Z, 1)^T$ in the world coordinate frame is mapped to the point $\boldsymbol{x} = (x, y, 1)^T$ in image coordinates by the point imaging equation:

$$\boldsymbol{x} = P\boldsymbol{X} \qquad (8)$$

Given a POM grid with resolution r (distance between to discrete locations) and a detection at the grid position (x_g, y_g), the 3D location (in homogenous world coordinates) of the center of the head would be at:

$$\boldsymbol{X_h} = (rx_g, ry_g, h, 1) \tag{9}$$

where h is the distance from the center of the head to the ground. So we can expect the head to be at the image location:

$$\boldsymbol{x_h} = P\boldsymbol{X_h} \tag{10}$$

If we know the height h' of the person, then $h = h' - k$ (k being the distance from the center of the head to the top of the head) and we can calculate the exact position of the center of the head in the image by the above equation. It is then enough to look around this position to detect a face.

Of course the exact height of the person is not known, and we should account for a possible error of the detection. Considering the average height of a person and allowing an $M\%$ margin for taller/shorter persons, we can search in a small rectangle of the image for a face. This is the equivalent to projecting a cube, that contains the head of a person with height $h' \pm M\%$, in the real world to a 2D rectangle in the image. For the rest of the chapter we refer to the search rectangle around the head as head ROI (Region Of Interest).

3.5.2 Simplifying the head location estimation

We can simplify the estimation of the ROI by exploiting the information in the POM configuration file. Inside the configuration file are defined all the rectangles standing for an individual at a specific grid position viewed from a certain camera. As explained in [34], they are generated similarly in the way that was described in the previous section. So instead of calculating the ROI every time using the camera projection matrix, we extract the upper part of the POM rectangle, again allowing an $M\%$ margin for detection error and shorter/taller people, since the POM rectangles where generated for a person of average height. A screenshot showing the POM rectangle and the head ROI is shown in Figure 14.

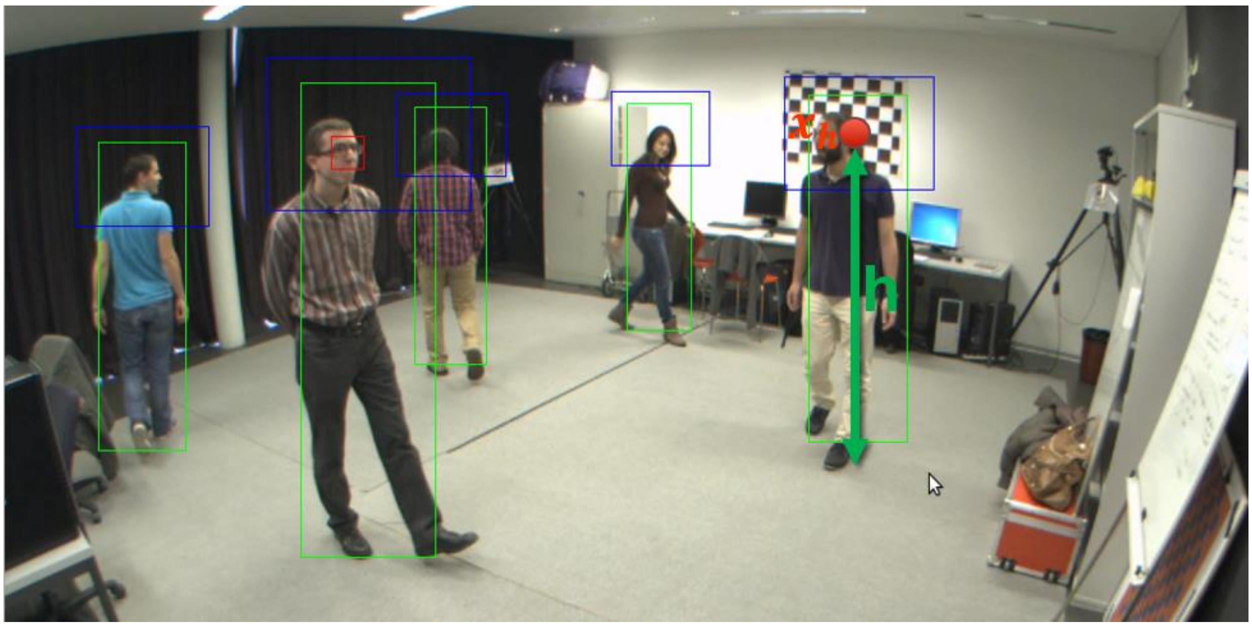


Figure 14 - Face detection with BBF exploiting POM and calibration information. Green boxes denote the positions where people are detected. Blue boxes are the extracted ROIs and red boxes are the face detections. The expected position of a head x_h is shown by the red dot.

3.5.3 Final form of the detector

While the above described method eliminates many false positive there are still cases where the detector is triggered by mistake, as the one shown in Figure 15. In both cases, the detections are within the ROI of the head but obviously they are wrongly classified as faces.

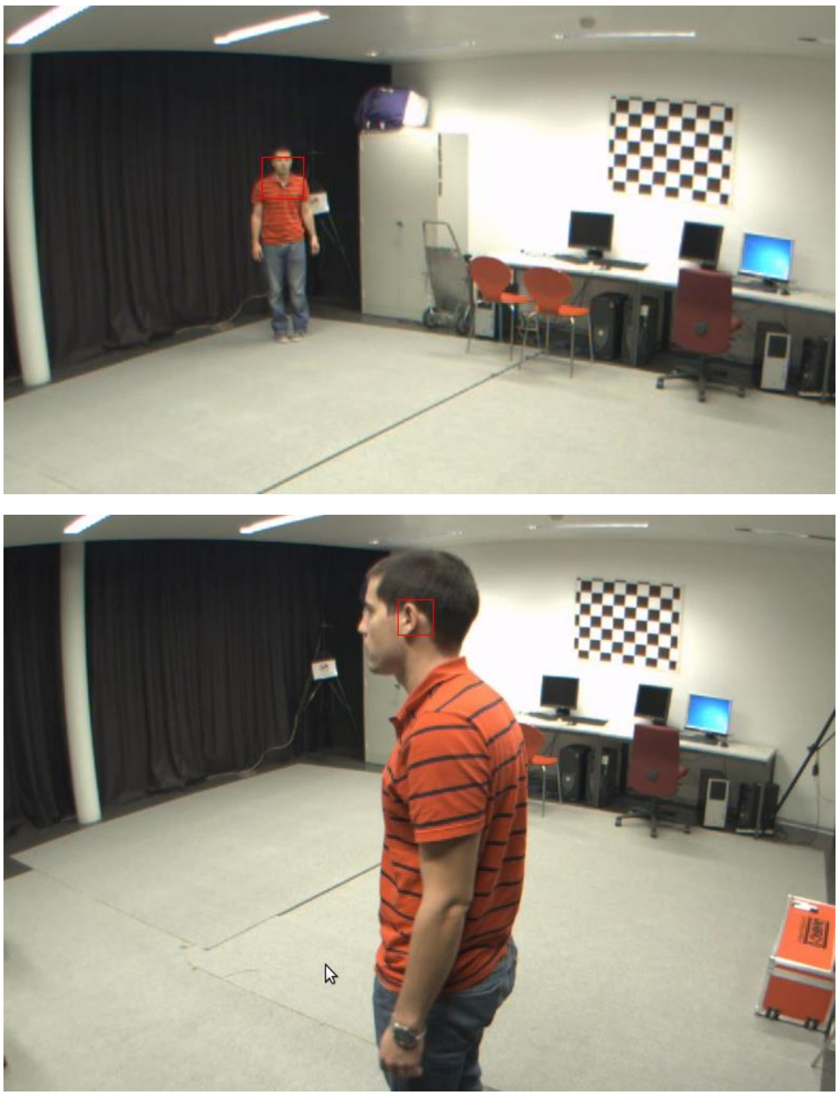


Figure 15 – Two examples where BBF + POM method fails by generating false positives. Obviously the size of the detected face is too large (and small respectively) to appear so far (and close respectively) from the camera. This is solved with the 3 rules introduced in the final form of the detector.

So the following steps take place in order to eliminate the false positives:

1. Search only in a region of interest (ROI) around the expected head position
2. Reject detections within ROI that are larger/smaller than the expected head size
3. Restrict the maximum number of detections within the ROI to one

We compare the performance of the original *bbf-opt* (see 3.4) face detector and the improved one, as discussed in this section. The results are shown below (Figure 16 and Figure 17).

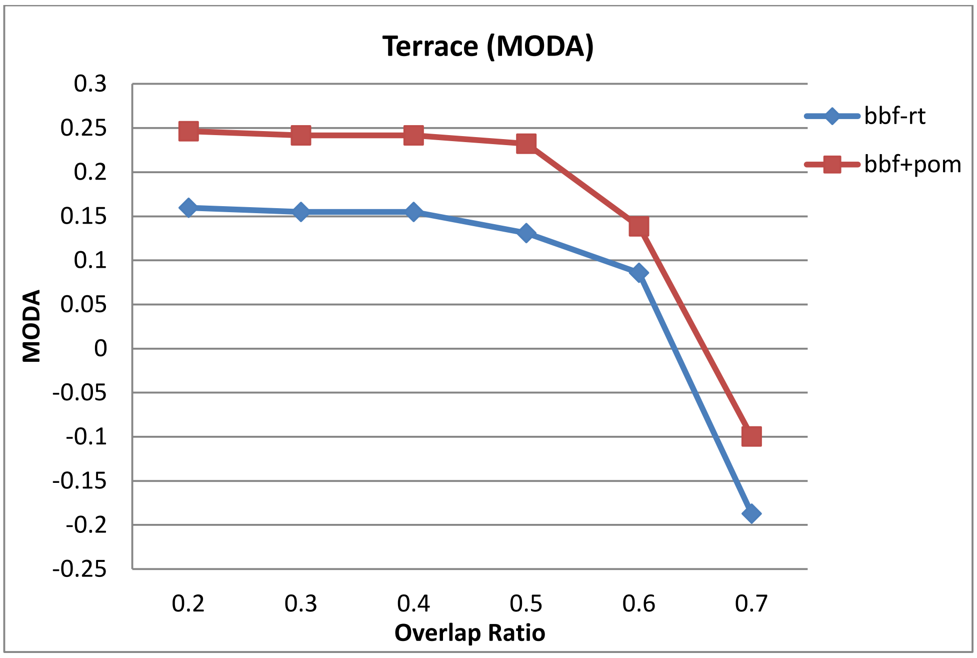


Figure 16 - Comparison of the final form of the face detection algorithm (bbf+pom) against the original BBF implementation (bbf-opt). Obviously the final detector is better because it eliminates most of the false positives.

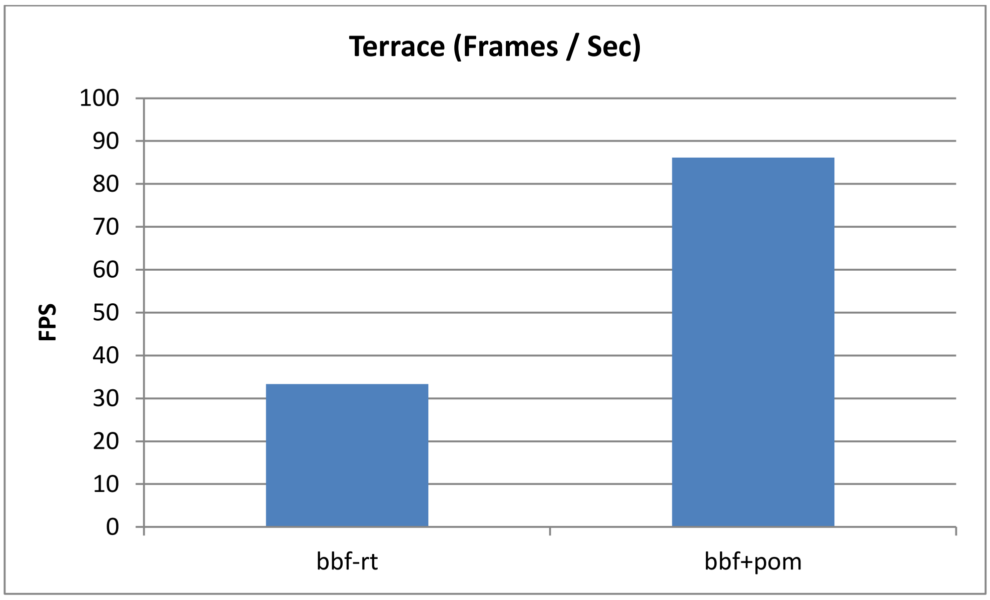


Figure 17 - Speed comparison of the final form of the face detection algorithm (bbf+pom) against the original BBF implementation (bbf-opt). Since the final detector prunes most of the image by searching only in places where heads are expected to be found, it is much faster than the original version.

The final version of the face detector, as expected, performs better since it produces less false positives. Also, since it only searches for faces in small areas of the image, it is about 2.5 times faster than the original version achieving detection at a rate of 85 fps on the Terrace sequence (for a single camera).

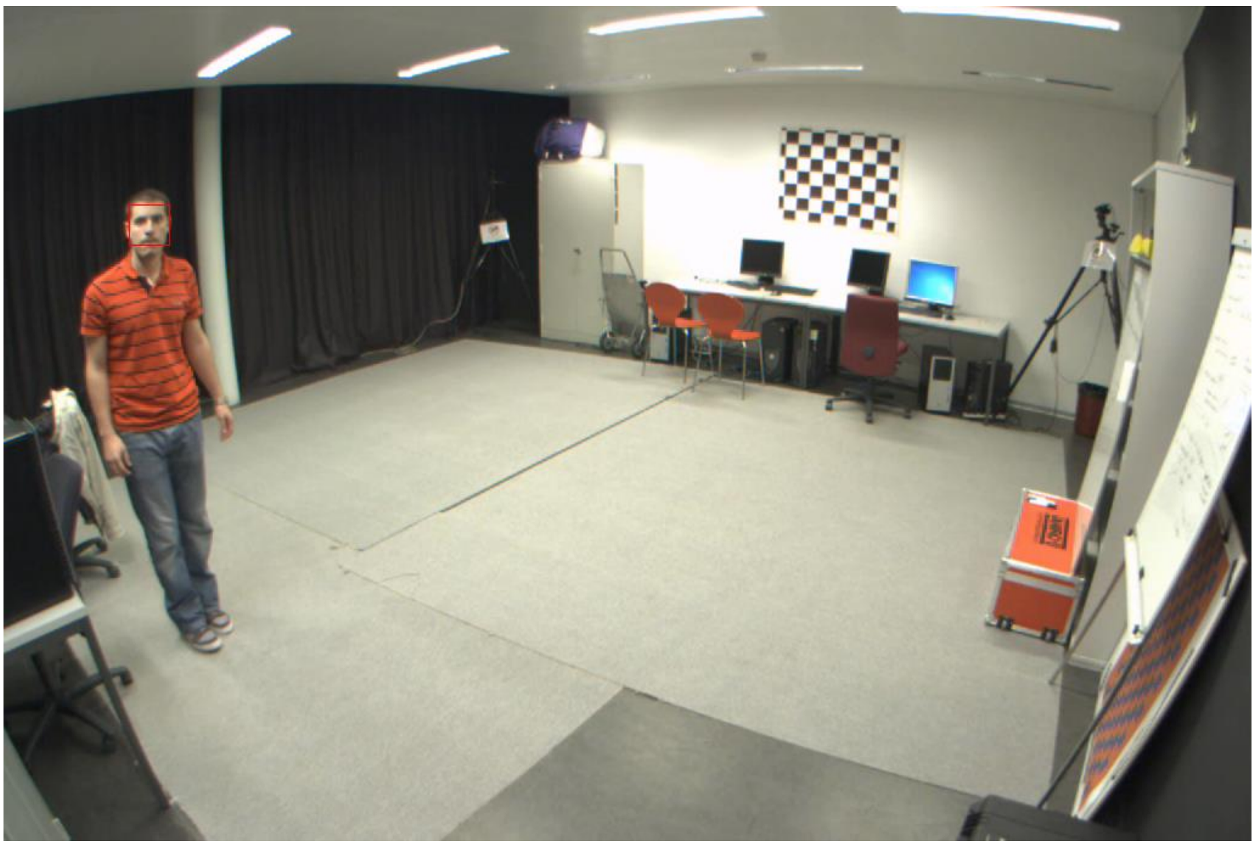


Figure 18 – A detection example of the final BBF + POM face detector.

The main objective of the face detector is to eliminate the false positives as they would interfere with face modelling and recognition. So we target for higher precision, rather than recall.

4. FACE MODELLING AND RECOGNITION

The modelling and recognition of the faces in our framework is based on the Local Binary Patterns that are presented on the first section of this chapter. As discussed in the literature review (section 2.2), the field of face recognition is pretty mature with hundreds of different methods proposed the last years. We choose the Local Binary Patterns descriptors for 3 reasons: a) Their performance, even though it's not the state-of-the-art, is very good, b) They are really simple and fast to compute, so they fit our goal of a real-time system, c) They are robust with respect to illumination, facial expression and alignment in contrast with other widely used algorithms.

4.1 Local Binary Patterns

The LBP operator was originally proposed in [40] as a texture classification feature. In its original form the feature is calculated at a pixel location (x_c, y_c) of the image I as follows:

$$LBP(x_c, y_c) = \sum_{n=0}^{7} 2^n \left\{ I(x_c, y_c) < I\left(x_c^{(n)}, y_c^{(n)}\right) \right\} \qquad (11)$$

where $\{ A \} = \begin{cases} 1, & if\ A\ is\ true \\ 0, & if\ A\ is\ false \end{cases}$ and $\left(x_c^{(n)}, y_c^{(n)}\right)$ denotes the nth neighbor of the pixel at (x_c, y_c). A schematic representation of the 8 neighbors, with their index, is shown in Figure 19. The first neighbor ($n = 0$) is considered the top-left and their index is incremented clockwise.

$\left(x_c^{(0)}, y_c^{(0)}\right)$	$\left(x_c^{(1)}, y_c^{(1)}\right)$	$\left(x_c^{(2)}, y_c^{(2)}\right)$
$\left(x_c^{(7)}, y_c^{(7)}\right)$	$(\boldsymbol{x_c}, \boldsymbol{y_c})$	$\left(x_c^{(3)}, y_c^{(3)}\right)$
$\left(x_c^{(6)}, y_c^{(6)}\right)$	$\left(x_c^{(5)}, y_c^{(5)}\right)$	$\left(x_c^{(4)}, y_c^{(4)}\right)$

Figure 19 – The 8 neighbors and their indexes, used for the computation of $\boldsymbol{LBP(x_c, y_c)}$

The idea is that every one of the eight neighbors is thresholded with the value at the center $I(x_c, y_c)$ so that an 8-bit number is generated which is then converted to an integer in $[0,255]$ which is the $LBP(x_c, y_c)$. Figure 20 illustrates this concept. The green values indicate pixels whose value is greater than the center pixel and red values indicate those pixels whose values is less than or equal to the center pixel. The 8-bit value is generated by concatenating the resulting binary numbers clockwise, starting on the top-left neighbor.

The feature is evaluated at each pixel location and an image $\hat{I}$ is generated with the computed LBP values. Finally a histogram of the 256 possible values is built that can be used for classification.

More formally the histogram can be defined as:

$$H(i) = \sum_{x,y} \left\{ \hat{I}(x, y) = i \right\}, \qquad 0 \le i \le 2^8 - 1 \qquad (12)$$

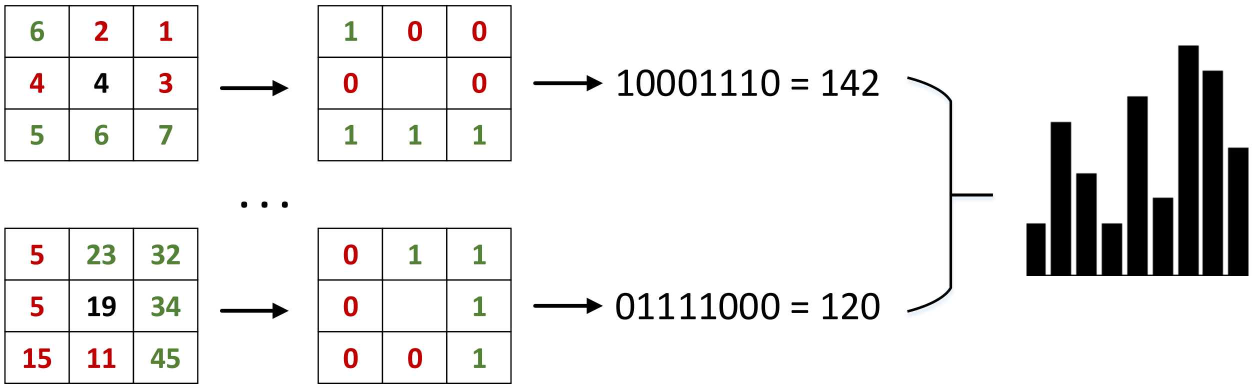


Figure 20 – The LBP computation steps. The following procedure is performed for each pixel in the patch. First the neighbors are thresholded based on the value in the centre. Those in green denote values greater than the center pixel and those in red denote less than or equal to the value in the center. The result of the thresholding is either 0 or 1 for each neighbor. Concatenating those numbers clockwise (starting at top left corner) gives an 8 bit binary number. This is converted to a decimal number in [0, 255]. When this is done for all pixels, a histogram of the numbers is created which is the LBP histogram of the image.

The idea was later extended [41] by *Ojala et al*. to allow different number of neighbors. The idea remained the same but instead of having a 3×3 neighborhood with 8 neighbors, there can be an arbitrary number of pixels P taken into account on a radius R of the central pixel. The notation (P, R) is used to define such a neighborhood. So formula 11 now becomes

$$LBP_{(P,R)}(x_c, y_c) = \sum_{n=0}^{P-1} 2^n \left\{ I(x_c, y_c) < I\left(x_c^{(n)}, y_c^{(n)}\right) \right\} \tag{13}$$

and there are 2^P different bins in the generated histogram which is defined formally as:

$$H(i) = \sum_{x,y} \left\{ \hat{I}(x, y) = i \right\}, \qquad 0 \leq i \leq 2^P - 1 \tag{14}$$

The neighbors are sampled on the circle with center (x_c, y_c) and radius R:

$$x_c^{(n)} = x_c + R \cos\left(\frac{2\pi n}{P}\right)$$
$$y_c^{(n)} = y_c - R \sin\left(\frac{2\pi n}{P}\right) \tag{15}$$

An example of $(8,2)$ neighborhood is shown in Figure 21. As can be seen, we might need to interpolate the values at the location $(x_c^{(n)}, y_c^{(n)})$. If this is required, bilinear interpolation is performed (any other interpolation scheme can be used).

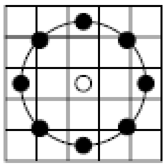


Figure 21 – A circular (8, 2) LBP neighborhood (image from [19]). The white dot is the pixel whose LBP value is calculated and the black dots are the neighbors. Interpolation is used to find the value of those positions.

If we select a coordinate system in which the four known pixel points are in the vertices of the square $(0,0) \times (1,1)$ then the value of the image I at an arbitrary position (x, y), according to the bilinear interpolation can be approximated by:

$$I(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} I(0,0) & I(0,1) \\ I(1,0) & I(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \qquad (16)$$

The resulting features are gray-scale invariant and can easily be transformed to rotation invariant if we shift (clockwise circular) the binary code as many times needed so that the maximal number of most significant bits is zero.

A local binary pattern is called uniform and denoted as LBP^{u2} if the number of transitions from 0 to 1 or vice versa is at most two. The authors of the paper, noticed on their experiments that uniform patterns with a neighborhood $(8, 1)$ account for more than 85% of all the occurrences. Considering that, it makes sense to use one bin for all non-uniform patterns which leads to a drastic decrease in the dimension of the histogram. Instead of 256 labels required for standard $LBP_{(8,1)}$, only 59 are required for uniform $LBP^{u2}_{(8,1)}$ without any decrease on the recognition performance.

4.2 Face modelling

Ahonen et al. used the Local Binary Patterns for the purpose of face recognition [19]. They exploit both the shape and the texture information. Given an input image I of a face, their algorithm for generating the descriptor of the face can be summarized as follows:

```
1:  Define a grid of $g_x \times g_y$ cells and divide $I$ accordingly
2:  For each cell $C$ do the following
3:     For each pixel $(x, y) \in C$ calculate the $LBP_P$ value $\hat{I}(x, y)$ as in (13)
4:     Compute the histogram of the patch $H_C$ as in (14)
5:     Normalize $H_C$
6:  Concatenate the histograms of all cells into the descriptor $H_I$
```

Algorithm 1 – Generating an LBP face descriptor

Each histogram is a vector of size 2^P and there are $g_x\ g_y$ cells so the final descriptor is a vector of size $S = 2^P g_x g_y$ (for standard LBP; uniform LBP patterns require less).

The texture of the facial areas is locally captured by the LBP operator while the shape of the face is encoded by the concatenating in specific order (left to right, top to bottom) the local LBP patterns. The combination of the texture and shape encoding make this method efficient, yet really fast. It's also robust to different facial expressions and lighting changes.

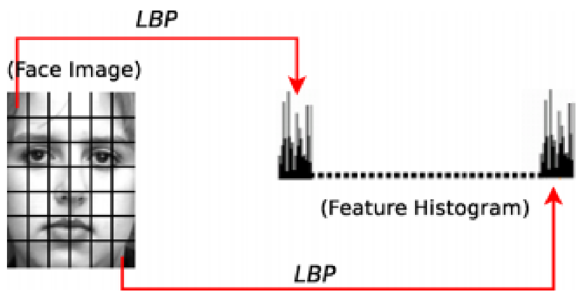


Figure 22 - LBP Feature descriptor. The image is split to a grid and the LBP histogram of each grid cell is computed. Those histograms are concatenated to create the final face descriptor (image from [42]).

We use the uniform $LBP^{u2}_{(8,1)}$ patterns and a 4×4 grid, leading to feature vectors of size 944. Once the descriptor H_I of image I is available, any classifier can be used to train a model and predict the class of a query image. The various classifiers that were used for the recognition are presented in section 4.4.

To model a person's face, we record him walking in the room for 2 minutes and capture multiple instances of his face as shown in Figure 23. Each detected face is encoded as an LBP histogram and the set of all histograms is the person's model.

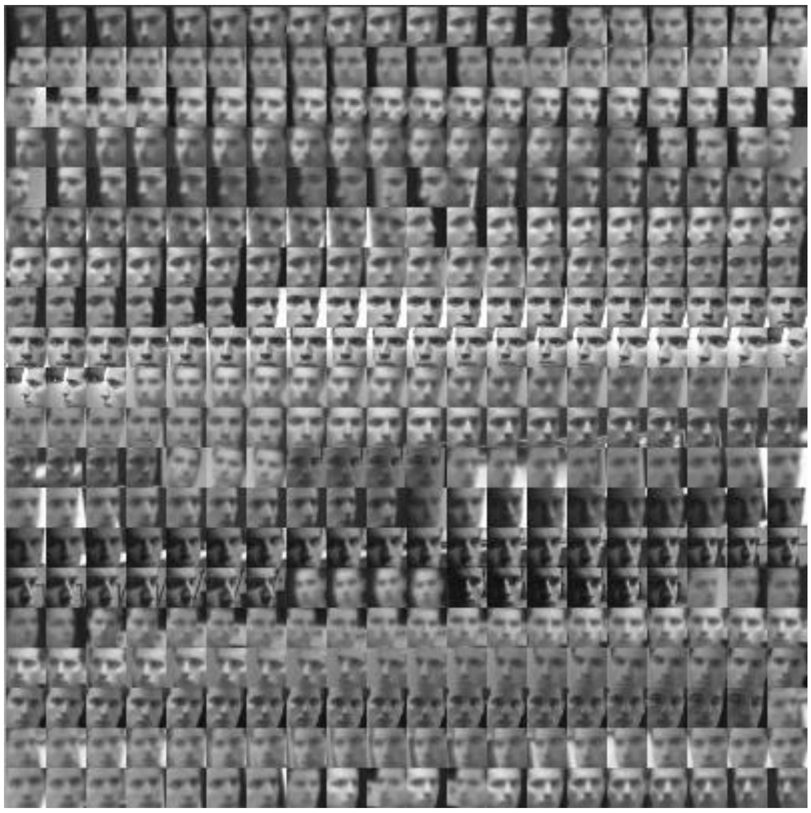


Figure 23 – Patches of a face model. There is a great variation in lighting and head pose of the captured person.

As can be seen in the figure, there is a great variance in the lighting conditions and the pose of the face which make the recognition robust to head pose and facial expression. The patches captured by one camera during modelling, can be used during the classification to match a face in another camera, effectively transferring the face model between cameras. The 3 rules introduced in the final form of the face detector (see 3.5.3) eliminate almost all false positives (non-faces) from being added to the face model.

4.3 Database of faces

The model M of each person, containing the LBP feature for each of the patches, is saved on disk. Every person is given a unique label that can be used to identify him. A multi-class SVM model is trained from those features and saved offline. The face recognition application loads this model and queries the classifier to match a face. Our database consists of 29 people, each of which has a model of size $|M| \approx 350$ features.

4.4 Face recognition

Given an unknown LBP feature vector any machine learning algorithm can be used to classify it against those on the database. As already mentioned, a multi-class SVM is employed to perform the classification task. It can quickly classify a new feature vector despite the large number of dimensions (944 in our case) and the number of instances of each class (around 350). After experimenting with different kernels for the SVM, the Radial Basis Function (RBF) is chosen. The RBF kernel for two vectors $\boldsymbol{u}, \boldsymbol{v}$ is defined as:

$$RBF(\boldsymbol{u}, \boldsymbol{v}) = e^{-\gamma|\boldsymbol{u}-\boldsymbol{v}|^2}, \qquad \gamma > 0 \tag{17}$$

To find the best values for the parameters γ and C (penalty parameter of the error term of the optimization problem that SVM is effectively solving) we search on a logarithmic grid and perform a 5-fold cross validation on the training set. The best parameters are then used to train the complete model. The contours of cross-validation accuracy for various (γ, C) values is shown in Figure 24.The accuracy of the 5-fold cross validation using the best (γ, C) parameters for our 29 class database is 99.3833%.

We use the SVM extension of [43] to predict not only the class where a sample belongs to, but also the probability estimates of belonging to each of the classes. The probability corresponding to the predicted label is used as the confidence score.

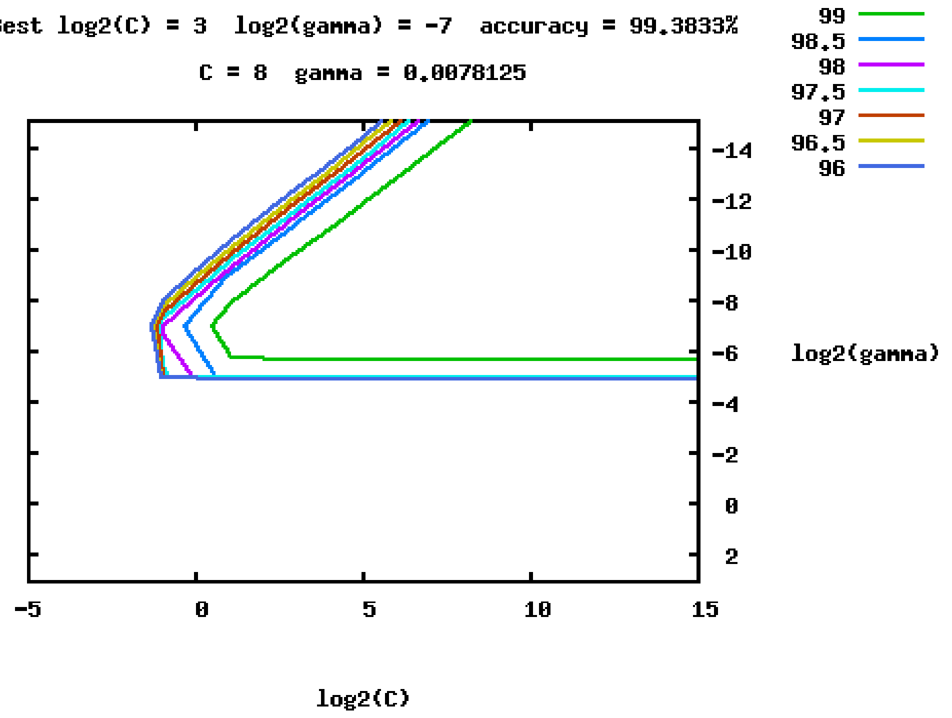


Figure 24 - SVM 5-fold cross validation accuracy contours for various values of (γ,C). The accuracy of the green contour reaches almost 100%.

Since the goal is to use the face recognitions to maintain people's identities while they are being tracked, we only need sparse face information; but when the information is available, it should be reliable. As in the face detection part, we care to eliminate the false positives (wrong recognitions in this case). To that end, we only keep a recognition if the probability of it belonging to the predicted class is above a threshold T. As we decrease the threshold we get more recognitions, but they might not be reliable. We choose $T = 0.8$ for our experiments, which gives very good recognition results (Figure 25).

The LBP histogram features are robust to illumination changes and facial expressions, while the SVM classifier is trained on face patches of various different poses of a head making the final classifier also robust to head pose.

While the SVM is used in the final version, two more classifiers were also tested; an 1-NN and a 3-NN. Their performance was much worse than the SVM classifier and they also required more time for the classification. The dissimilarity measure used for those is the Chi-square statistic (x^2). It was shown in [19] that it outperformed other metrics for face classification using the LBP histogram features. Given two images I, J and their respective descriptors $\boldsymbol{H_I}, \boldsymbol{H_J}$ of size S, the chi-squared measure is:

$$x^2(\boldsymbol{H_I}, \boldsymbol{H_J}) = \sum_{i=0}^{S-1} \frac{\left(\boldsymbol{H_I}(i) - \boldsymbol{H_J}(i)\right)^2}{\boldsymbol{H_I}(i) + \boldsymbol{H_J}(i)} \tag{18}$$

4.5 Experiments

We tested the 3 classifiers (1-NN, 3-NN, SVM) on the 9 people dataset described in 5.3.1. The sequence are 4 minutes long and are recorded by 6 cameras at 1032 x 778 pixels with a frame-rate of 30 FPS. There are 9 people in the scene, all of which are stored in the database (total 29 people in the database). First we employ the face detector of 3.5.3 to acquire a total of 2454 potential face images. Those images were hand labeled and the 3 classifiers were evaluated on this set. For each image the classifiers outputs the predicted class and the confidence value. By setting a threshold on the confidence value, as already discussed, we keep only the most reliable recognitions. Lowering the threshold, produces more recognitions that can be exploited by the tracker but they might not be so reliable. Figure 25 plots the recognition rate against the percentage of recognitions that are above threshold. The SVM achieves nearly perfect recognition rates while it requires around 40% less time to classify a face than the other two methods.

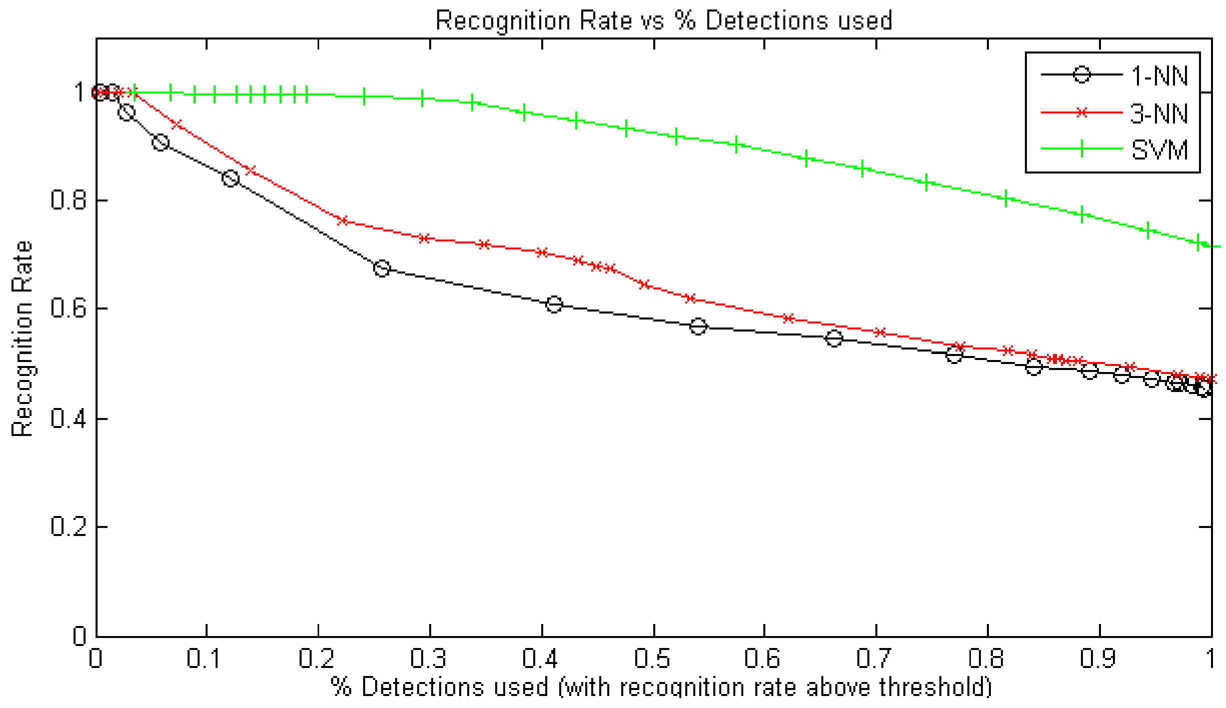


Figure 25 – Comparison of the 3 classifier methods. The recognition rate is plotted against the percentage of face detections that are used for recognition. As the threshold is lowered, more recognitions are considered reliable but the recognition rate drops. As is obvious by the diagram, the SVM method gives nearly perfect results and can exploit almost 30% of the face recognitions without any performance degradation.

5. INTEGRATION WITH PEOPLE TRACKER

The end goal of this thesis is to integrate the face recognition with a people tracker. So we exploit the sparse recognitions to: a) identify the people in the scene and b) infer identity-preserving trajectories with less identity switches in crowded scenes. We integrate the face recognitions with two trackers developed in the CVLab.

5.1 Real-time Multi-Object Kalman Filter Tracker

The Real-time Multi-Object Kalman Filter Tracker [44], simply called Kalman tracker for the rest of this thesis, was implemented last year as a semester project. The ground plane is discretized on a grid and the input to the algorithm is a Probabilistic Occupancy Map (POM) that contains the probabilities of presence of individuals in each grid cell. Using those, potentially noisy, detections the algorithm finds the best matching between the detections and the objects being tracked by solving an optimization problem using the Hungarian algorithm (Appendix B). A Kalman filter [45] is employed to keep track of each person's motion on the ground plane. Appearance information, in the form of a color histogram in the YCbCr color space, is exploited to improve the detection – object linking step. The Kalman tracker works on a frame by frame basis so the resulting trajectories might not be globally optimum. However it's very efficient and can run in real time.

5.1.1 Incorporating the face recognitions

The face information is used in two ways by the Kalman tracker. Since the Kalman tracker is a frame-by-frame method, if no information is available for some frames it might lose track of the object. When, later, a face is recognized for a detected person on some location of the grid, the Kalman tracker for that person can re-initialize from that point and continue tracking the individual. The second way that the face information is exploited targets to generating identity-preserving trajectories. Each person being tracked maintains a score S_l for every identity label l. When a face is detected and recognized with confidence c_l and the detection is linked with a person, then the identity scores for this person are updated as shown below:

$$R_l = \frac{c_l + R_l^{t-1} * D^{t-1}}{1 + D^{t-1}} \ , \qquad \mathrm{S_l} = \frac{R_l}{D} \tag{19}$$

where R_l denotes the updated average confidence of this person for label l, and D is the number of face recognitions that have been assigned to this person. The $(t-1)$ superscript denotes the previous value.

If the highest identity score for that individual (for some label l) is above a threshold, then the identity l is given to that person. While the highest score is below the threshold, the tracked person is considered unknown.

If at some point during the video a person is recognized with an identity different than the one that he is assigned to, then he is marked as being in a possible swap mode. When two people A, B are in possible swap mode, their most recent recognitions are checked. If person A was identified as B and vice-versa then their identities are switched. This is extended to more than two people by checking for a circle in the recent recognitions.

5.2 Multi-Commodity Network Flow Tracker

The Multi-Commodity Network Flow (MCNF) Tracker is an extension of the K-Shortest Paths (KSP) method [31]. The ground plane is represented by a discrete grid and at each frame a

Probabilistic Occupancy Map (POM) [34], containing the probabilities of presence of individuals in each grid cell, is given.

Given the noisy detections in individual frames, one can link detections across frames to come up with trajectories. When dealing with many objects, the linking step results in difficult optimization problem in the space of all possible families of trajectories. The KSP method [31] formulates the linking step as a constrained flow optimization problem that leads to a convex problem. Due to the problem's structure, the solution can be obtained by running the k-shortest paths algorithm on a generated graph. The k-shortest path algorithm reaches the solution in polynomial time.

The MCNF tracker [32] extends the formalism of KSP to account for individual identities. The linking step is formulated as a multi-commodity network flow problem on a Directed Acyclic Graph (DAG). This is an NP-complete problem that requires solving an Integer Programming problem, so it's relaxed to a Linear Programming one. People's trajectories are modeled as continuous flows going through an area of interest. Identities are assigned to each trajectory by appearance cues and different flows shouldn't be allowed to mix. A brief description of the method is given below.

Given the discretized ground plane, containing K cells, the probabilities of presence of people (POM) in each cell for all T frames and the number of identity groups L (where N_l is an upper bound to the number of people in group l), a Directed Acyclic Graph (DAG) of size $K \times T \times L$ is generated as follows. Every node represents a spatial location at a specific frame for a particular identity group. Edges between nodes represent admissible motion between locations. Since the number of tracked people may vary over time, with persons coming in or leaving from the tracking area, the framework allows for flows to enter or exit the graph. This is accomplished by the introduction of two virtual nodes (v_{source} and v_{sink}) that are connected with all the locations where an individual can enter or exit the scene. Also v_{source} is linked with the locations of the first frame and v_{sink} is linked with the locations of the last frame to allow tracking of people that are already in the scene when the algorithm starts/ends. A sample graph is shown in Figure 26.

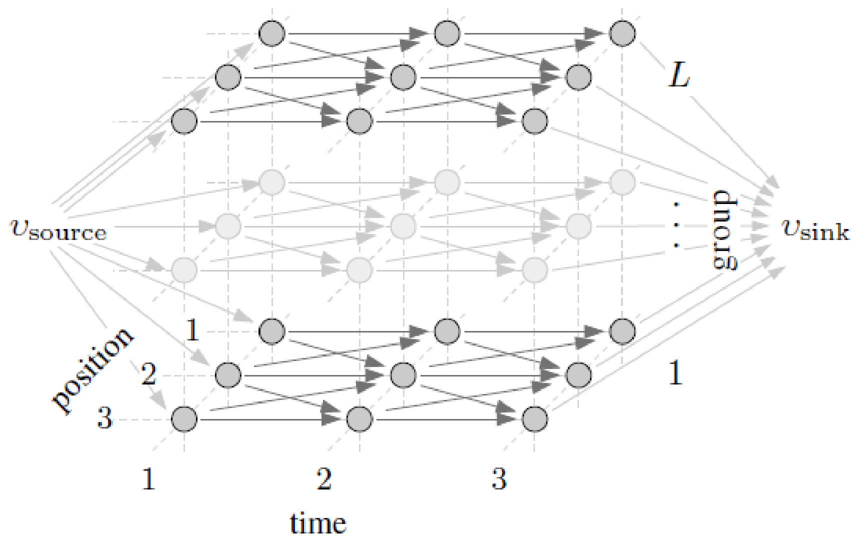


Figure 26 – MCNF Tracking algorithm formulated using a Directed Acyclic Graph. Each node represents a position of the grid, at a given frame for a specific identity group. Edges encode potential move of a person between locations in space / time (image from [46]). Two virtual locations v_{sink}, v_{source} are added to allow people to enter or exit the scene at specific grid positions. They are also linked to all the grid locations on the first and last frames to allow people to be in the scene when the algorithm begins and ends. The method encodes the movement of people as flows in the graph from v_{source} to v_{sink}.

Let $m_k^l(t)$ be a variable associated with every node that denotes the number of people of identity group l standing at location k on frame t and $f_{i,j}^l(t)$ be a variable associated with every edge and denotes the number of people from group l moving from node i to node j at time t. Also, let $p_k(t)$ the probability of a person being at location k of the grid at time t and $\varphi_k^l(t)$ the probability that the identity of a person occupying location k at time t is l. Finally let $N(k) \subset \{1, \dots, K\}$ be the neighborhood of k, which are the locations that can be reached from k in one time instant.

Skipping the details (the reader is referred to [32], [46] for more information), the set of physically possible trajectories that best explained the observed images can be obtained by solving the following Integer Program with respect to the flows $f_{i,j}^l(t)$.

$$\begin{aligned}
&\text{Maximize} && \sum_{t,i,l} \log\left(\frac{p_i(t)\varphi_i^l(t)L}{1-p_i(t)}\right) \sum_{j\in N(i)} f_{i,j}^l(t) \\
&\text{Subject to} && \forall t,i \ \sum_{j\in N(i)} \sum_{l=1}^{L} f_{i,j}^l(t) \le 1 \\
& && \forall t,l,i \ \sum_{j\in N(i)} f_{i,j}^l(t) - \sum_{k: i\in N(k)} f_{k,i}^l(t-1) \le 0 \\
& && \sum_{j\in N(v_{source})} f_{v_{source},j} - \sum_{k: v_{sink}\in N(k)} f_{k,v_{sink}} \le 0 \\
& && \forall t,l \ \sum_{i=1}^{K} \sum_{j\in N(i)} f_{i,j}^l(t) \le N_l \\
& && \forall t,i,l,j \ f_{i,j}^l(t) \ge 0
\end{aligned} \qquad (20)$$

The above Integer Program is relaxed to a Linear Program by allowing the variables to become real-valued. While the result of the LP program might be a non-integer solution in some cases, this happened quite rarely on the experiments and the non-integer results are rounded. Because the full graph will be extremely large for long sequences and running the Linear Programming on it might be slow, a two-step process is implemented. First the KSP method [31] is run, this is the same as having $L = 1$ (single commodity network flow). This produces trajectories that may have identity switches but contain the spatial locations where people are expected to be found at any frame. So all other grid cells can be eliminated and the MCNF method is run on this smaller graph. A final modification to the method, is the introduction of the tracklets. Nodes that unambiguously connected are grouped into tracklets. Each tracklet corresponds to a single node in the graph. So an even smaller graph is obtained on which the Linear Program can be solved. The latter version of the algorithm is called T-MCNF while the non-tracklet version is denoted as MCNF.

5.2.1 Incorporating the face recognitions

To incorporate the face recognitions to the MCNF tracker is straightforward. We have L identity groups, one for each person in the database and one for the unknown people. The face detector (see section 3.5.3) searches for a face on the head part of each POM detection at a location k and frame t and the face classifier (see section 4.4) calculates the probability of a face belonging to an identity l, which is the probability $\varphi_k^l(t)$ appearing in the optimization problem. Since we want to use recognitions that we are really confident that are correct, we only keep those that are above a threshold, 0.8 in our case. When the face information is not present we set $\varphi_k^l = \frac{1}{L}$.

5.3 Experiments

5.3.1 Datasets

For the purposes of this thesis we recorded two new datasets in the CVLab demo room. Both datasets are recorded by 6 synchronized cameras at a resolution of 1032 x 778 pixels and a frame-rate of 30 fps. Four of them are in the corners having a clear view of all the room, the fifth one is zoomed in the central part of the room, while the sixth captures the area in front of the door as shown in Figure 27.

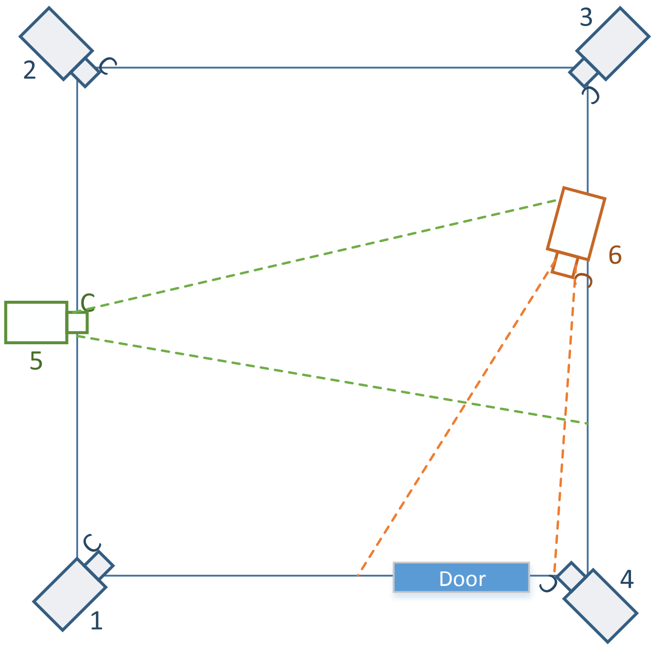


Figure 27 – A diagram of the recording room. There are six cameras. Four of them are in the corners of the room with a clear view of the whole scene while the other two are zoomed and capture the areas shown in dotted lines.

The first dataset consists of 3500 frames. There are 6 people coming into the room and walking around. The second dataset is more challenging and consists of 9 people coming in and out of the room at various times; at any single point there are from 1 up to 9 people in the scene. The videos consist of 7300 frames. Our face database contains 29 people, 5 out of the 6 people in the first sequence are in the database, while all 9 of the second sequence are in the database.

Three screenshot from the datasets can be seen in Figure 29 (view from camera 2), Figure 28 (view from camera 3) and Figure 30 (view from camera 5).

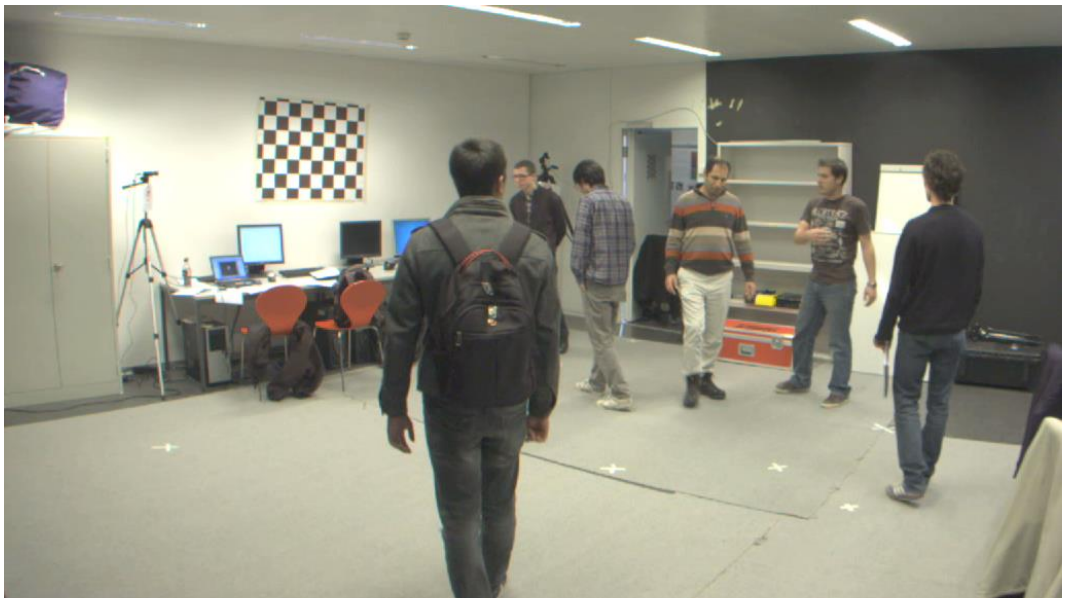


Figure 28 - Screenshot of the 6 people sequence capture from camera 2.

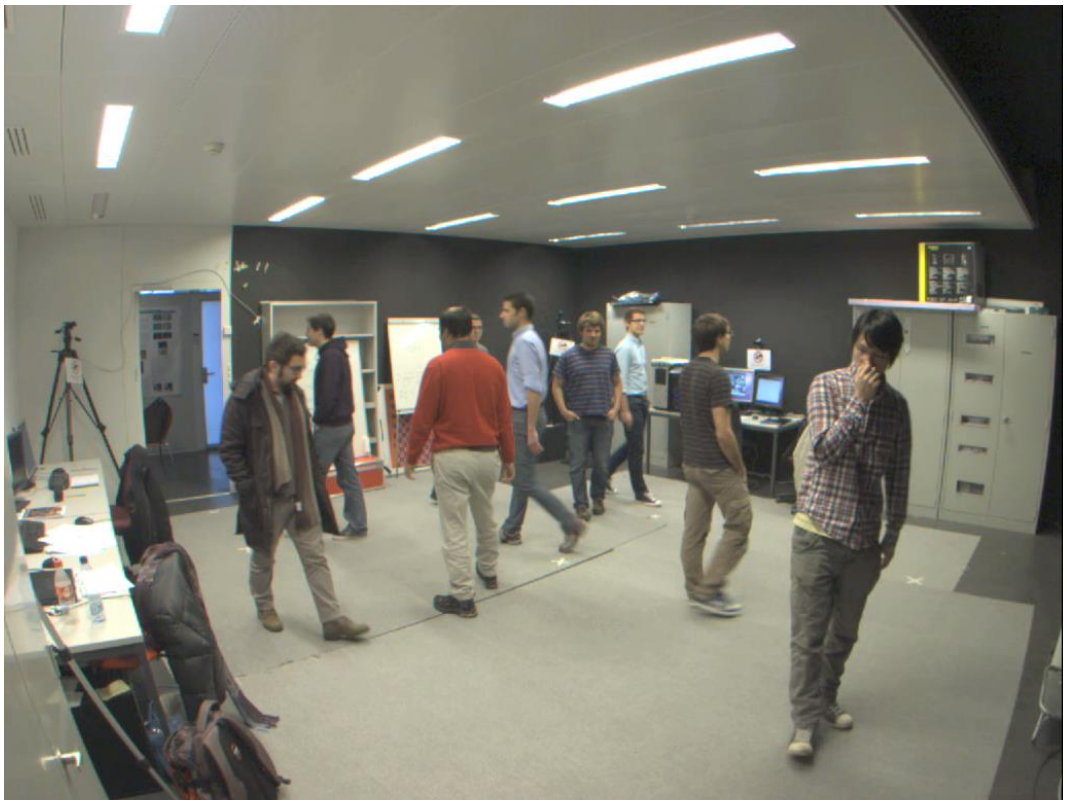


Figure 29 - Screenshot of the 9 people sequence captured from camera 3.

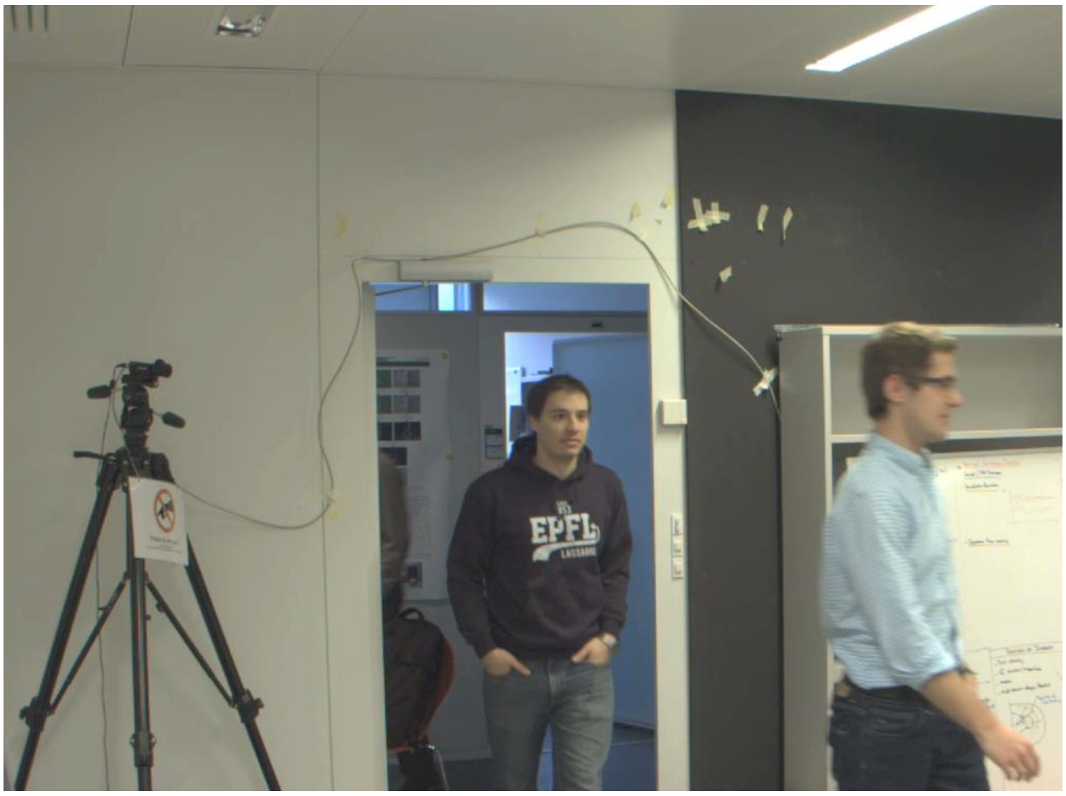


Figure 30 - The view from camera 5 that looks at the door.

5.3.2 Multiple Object Tracking Accuracy (MOTA) and GMOTA

A standard metric for object tracking evaluation is the Multiple Object Tracking Accuracy (MOTA) [38]. It is defined as:

$$MOTA = 1 - \frac{\sum_t \big(c_m(m_t) + c_f(fp_t) + c_s(mme_t)\big)}{\sum_t g_t} \tag{21}$$

where m_t is the number of missed detections, fp_t is the number of false positives, g_t is the number of ground truth detections and mme_t is the number of instantaneous identity switches. The c_m, c_f and c_s are weighting functions set to $c_m(x) = 1, cs(x) = 1$ and $cs(x) = \log_{10} x$, as in [46].

The GMOTA metric, defined in [46], is an extension of the well-known Multiple Object Tracking Accuracy (MOTA) metric, where the mme_t term is substituted by $gmme_t$. It is defined as:

$$GMOTA = 1 - \frac{\sum_t \big(c_m(m_t) + c_f(fp_t) + c_s(gmme_t)\big)}{\sum_t g_t} \tag{22}$$

The term $gmme_t$ measures the proportion of identity switches in a global manner. This makes GMOTA more appropriate for evaluating applications where identity preservation is crucial. The $gmme_t$ term is increased at every frame where the detection label doesn't correspond to the ground truth identity (unlike mme_t, in MOTA metric, which is increased only at the frame where the identity switch happened). More information about the definition of $gmme_t$ can be found in [46].

5.3.3 Results

The KSP algorithm gives almost perfect results, in terms of GMOTA score, on the 6 people sequence, thus leaving no space for improvement to the T-MCNF version. However the T-MCNF method can correctly identify all 5 persons that are stored in the database and assign the “guest” label to the 6th person. The identities of each person are maintained throughout the whole video without any identity switches. In the case of Kalman Filter, the use of face information (KALMAN-F curve) greatly improves the tracking results in terms of GMOTA score. Two more variations of the Kalman filter are also evaluated, one exploiting color information (KALMAN-C) and one exploiting both color and face information (KALMAN-FC). Those two perform equivalently to the method that uses the face recognitions. The comparative results are shown in Figure 31. The values in the x-axis denote the maximum allowed distance (in grid cells) between the tracked object and the ground truth so that a hit is counted.

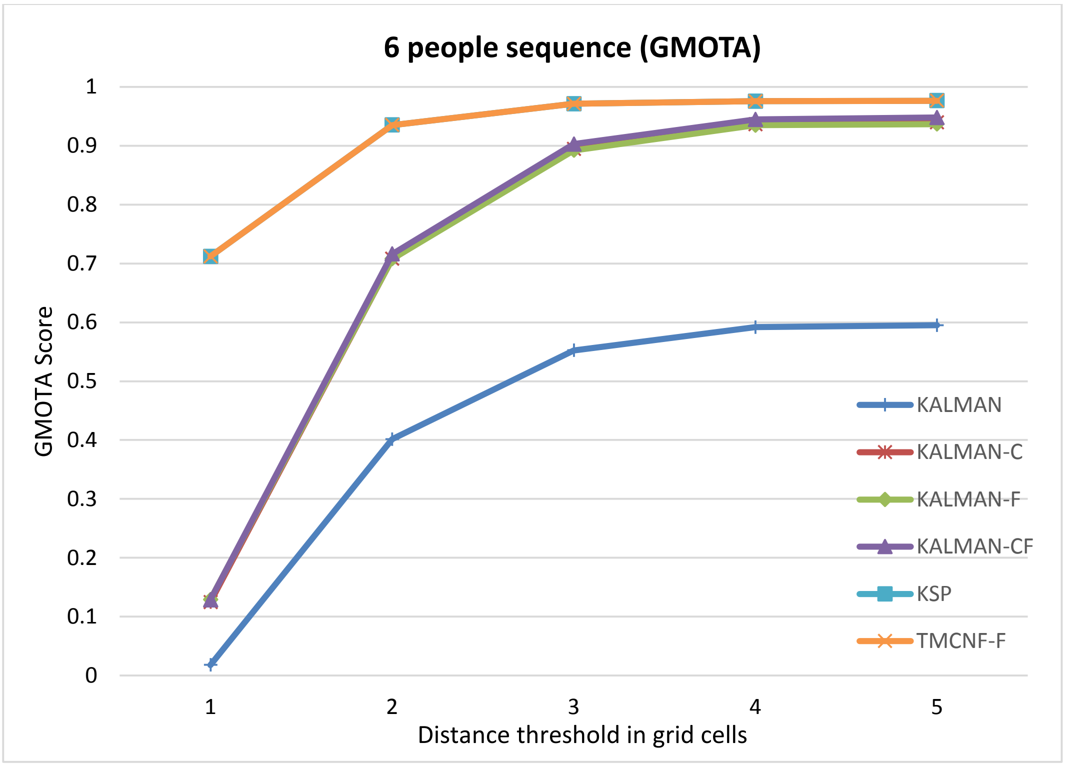


Figure 31 – GMOTA comparison of the tracking methods on the 6 people video sequence. The “C” versions denote the use of color model and the “F” versions denote the use of the face recognitions. The KSP solver produces excellent results leaving no space for improvement to the T-MCNF tracker. However the T-MCNF method exploits the face information and can correctly identify everyone in the scene throughout the video. For the Kalman filter, the use of face recognitions greatly improves the tracking results, as does the color model.

The 9 people sequence is more challenging. The GMOTA results can be seen in Figure 32. In this case the KSP method fails to generate identity-preserving trajectories and its performance in terms of GMOTA score is limited to around 50%. The T-MCNF algorithm drastically improves the performance of the KSP by exploiting the face recognitions. The sequence is proved to be hard for the Kalman filter, but still a small improvement in performance can be noticed by the introduction of the face information.

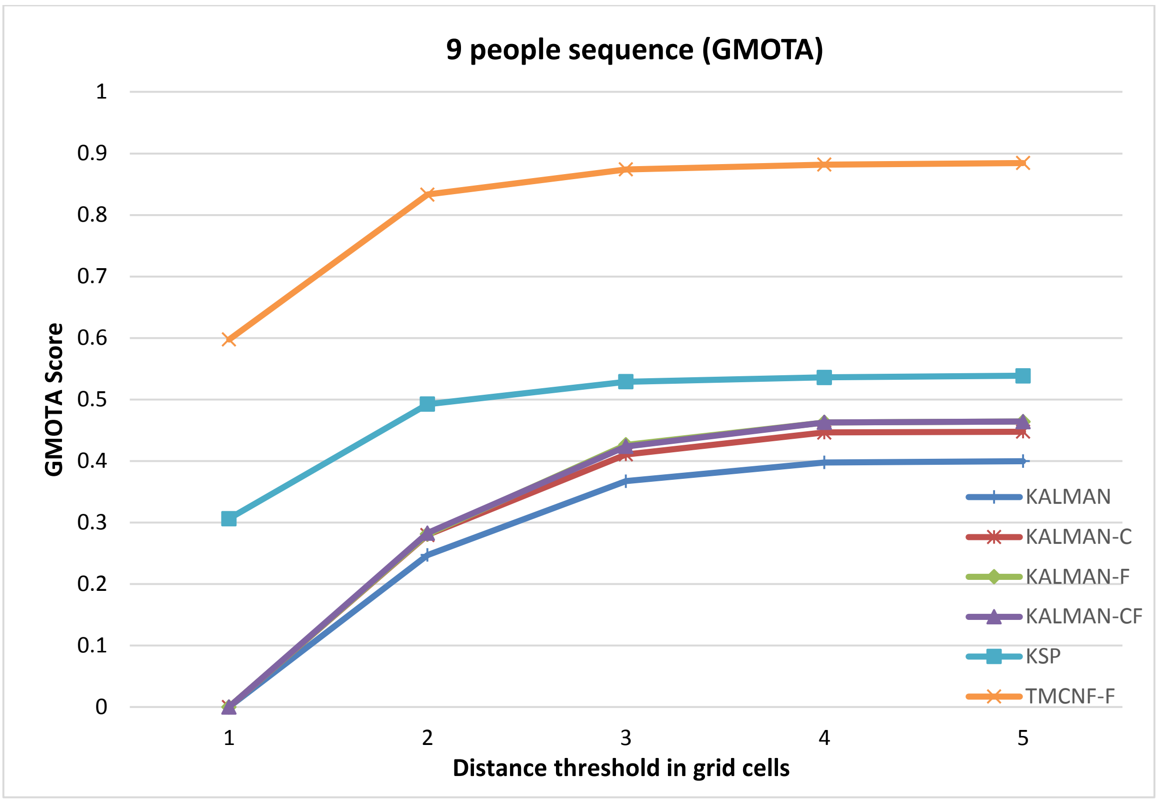


Figure 32 – GMOTA comparison of the tracking methods on the 6 people video sequence. The "C" versions denote the use of color model and the "F" versions denote the use of the face recognitions. The use of face information drastically improves the tracking results in the case of KSP and T-MCNF. For the Kalman filter, there is small improvement in performance by the use of the face / appearance information.

Some screenshots of the two video sequences overlayed with the tracking results are shown below. One person is marked as "guest" in Figure 33; his face model isn't stored in the database and consequently there are no consistent face recognitions associated with him during the tracking.

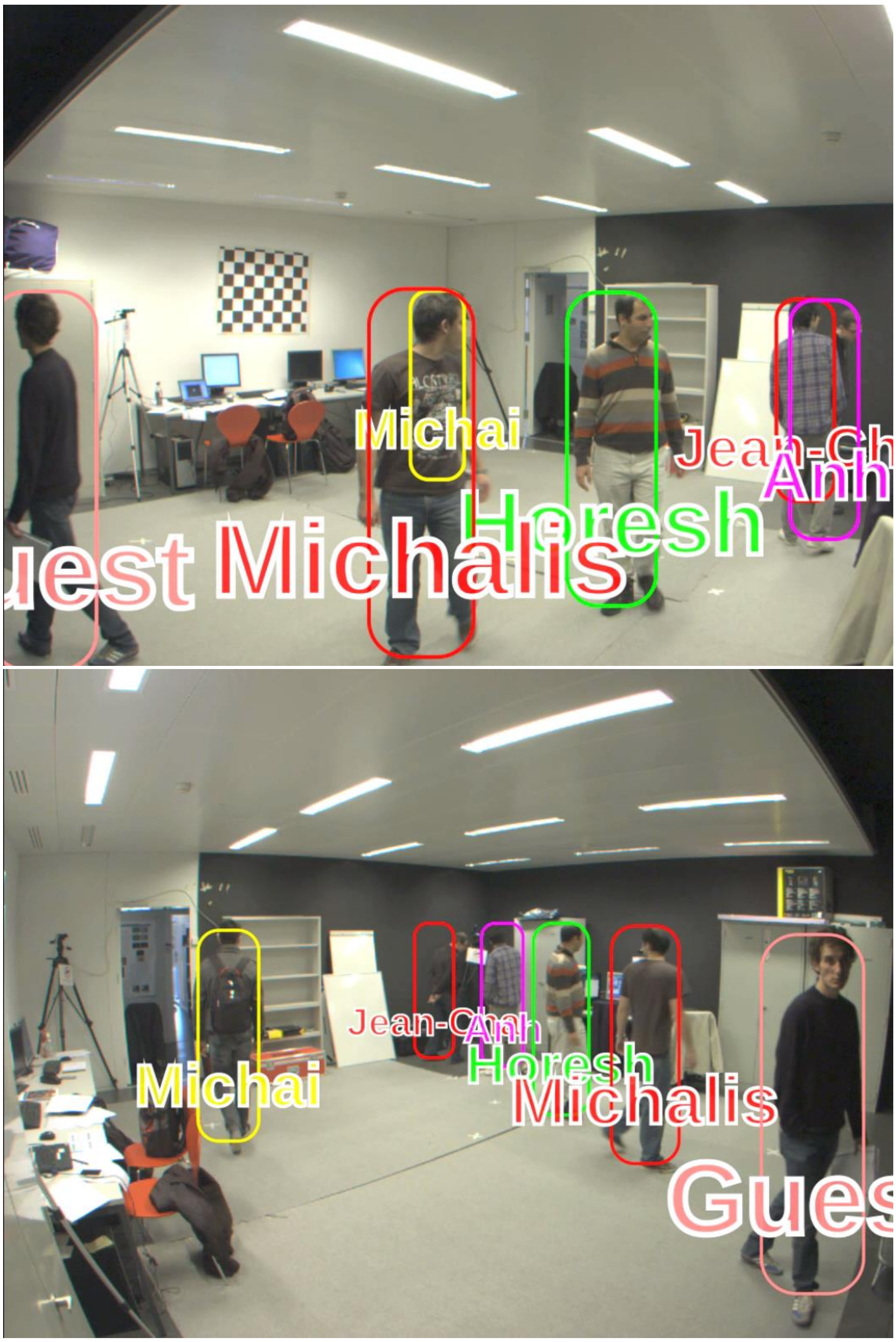


Figure 33 - Screenshot of the tracking results from 6 people video sequence as seen from cameras 2 (top) and 3 (bottom). Everybody in the scene is correctly identified. The person marked as a "guest" isn't saved in the database, consequently he isn't recognized throughout the video.

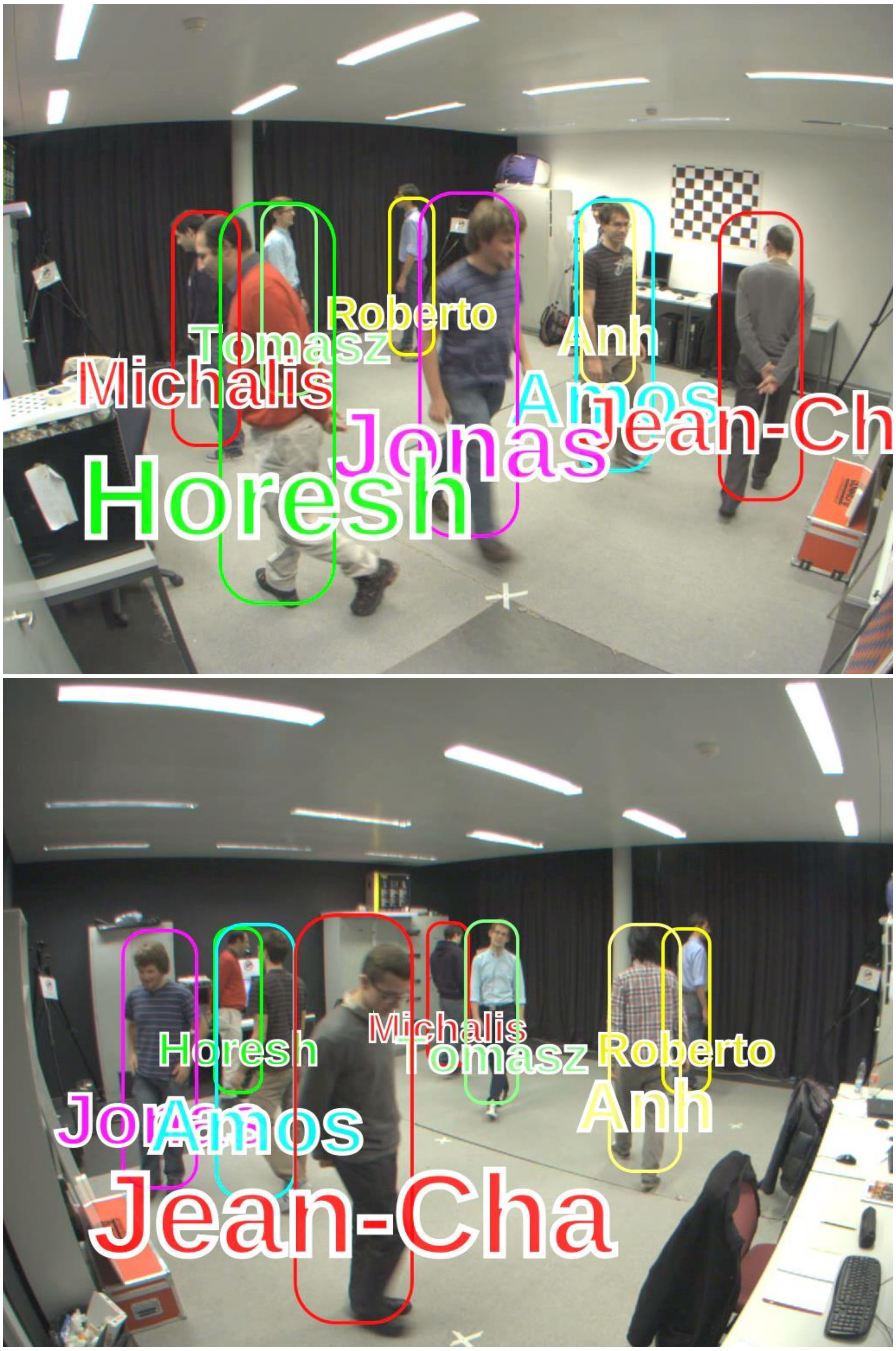


Figure 34 - Screenshot of the tracking results from 9 people video sequence as seen from cameras 1 (top) and 4 (bottom). Everybody in the scene is correctly identified.

6. IMPLEMENTATION DETAILS

All the system is implemented in C++ with standard libraries. The Probability Occupancy Maps (POMs) required as input, we use the publicly available POM software[1]. The K-Shortest Path tracker that is used for the first part of the T-MCNF method is also publicly available[2]. For all the SVM related operations, we used the LIBSVM [43] library[3]. In the following sections, we provide more implementation details for some parts of the system and the supporting tools that were created as part of this thesis. Individual applications for each task (Face Detection, Modelling and Recognition) were built as well as standalone applications that perform detection and tracking or detection and modelling together.

6.1 Web-based people annotation application

As part of this thesis, a web-based annotation application for multi-camera object tracking was created. It's implemented purely in HTML5 and Javascript. The input to the application is the frames of the videos of each camera, that can be stored either locally or on a remote website and the calibration files of the scene should be provided. A screenshot of the application is shown in Figure 35.

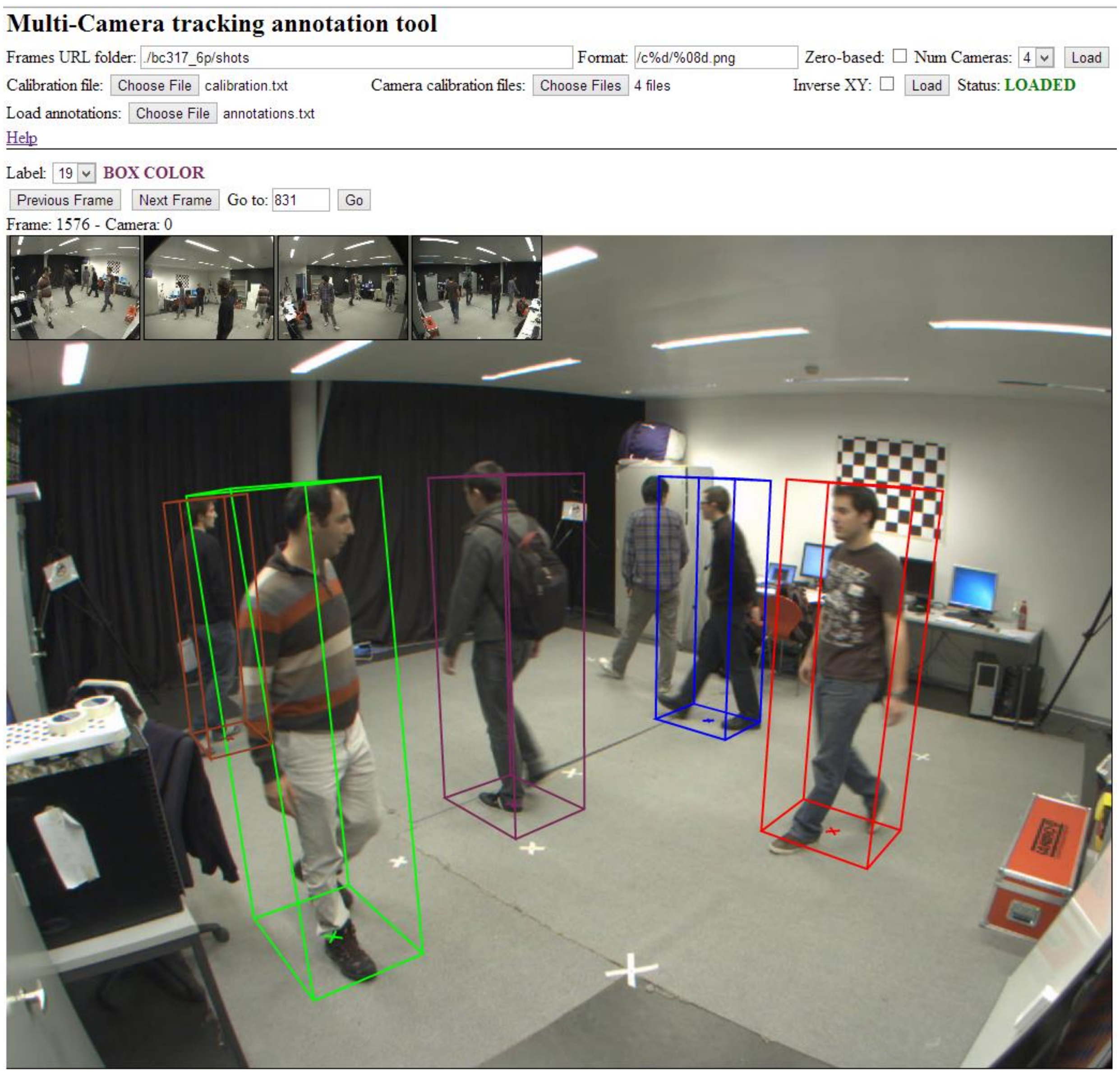
The user can annotate multiple people with different label / color, by selecting a point on the ground plane where the person stands. A bounding box is drawn on every view of the frame, so that the user can find the exact position of the person. The application can generate various output formats:

- World coordinates of each person at each frame
- Grid coordinates of each person at each frame
- Rounded grid coordinates of each person at each frame
- Grid cell id (as defined in POM software) of each person at each frame
- Trajectories in the format generated by the open source KSP software
- Ground truth format, used for evaluation of trajectories

Loading previously annotated data is also supported. All the annotations for the two new datasets were produced using this application.

The tool can easily be extended in the future to handle videos as input, instead of individual frame images.

[1] POM: http://cvlab.epfl.ch/software/pom/
[2] KSP: http://cvlab.epfl.ch/software/ksp/
[3] LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/



Multi-Camera tracking annotation tool

Frames URL folder: ./bc317_6p/shots Format: /c%d/%08d.png Zero-based: ☐ Num Cameras: 4 Load

Calibration file: Choose File calibration.txt Camera calibration files: Choose Files 4 files Inverse XY: ☐ Load Status: LOADED

Load annotations: Choose File annotations.txt

Help

Label: 19 BOX COLOR

Previous Frame Next Frame Go to: 831 Go

Frame: 1576 - Camera: 0

Figure 35 - Screenshot of the multi-camera tracking annotation software.

6.2 Face detection application

The three face detection methods presented in Chapter 3, were integrated to the system. The OpenCV implementation of the Viola – Jones was used. For the Binary Brightness Features, we used the libccv[1] library, while for the deformable detector we used the code created in the lab. The required detector can be easily selected by changing the configuration file. The default detector is the BBF. As described in 3.5.3, the face detector is then integrated with the POM people tracker to eliminate false positives and speed up the process. The application takes as input the POM detections and the video sequences and outputs the positions of each face in image coordinates for each camera.

[1] Libccv: http://libccv.org/

6.3 Face modelling application

The face modelling application of our framework, takes as input the videos of a single person from a number of cameras (6 in case of the CVLab demo room), the POM detections and the face detections as generated by the face detector. After processing the videos, a model of the face for the specific person is generated and can be later used for recognition. The model M is based on the Local Binary Patterns and the algorithm for its generation is shown below.

```
1.  $M = \emptyset$
2.  For each camera $f$
3.    For each camera $c$
4.      $d = detection(f, c)$
5.      $p = generatePatch(d)$
6.      If $(validate(p, M) = false\ OR\ size(d) < T)$ the discard $d$ and go to 3
7.      If $(|M| \geq MAX\_MODEL\_SIZE - 1)$ then
8.        $k = random(0,\ MAX\_MODEL\_SIZE - 1)$
9.        $M = M \setminus \{M[k]\}$
10.     $M = M \cup p$
```

Algorithm 2 - Face modelling algorithm

T is the minimum detection size required for face recognition (we use $T = 50$ in our experiments) and $generatePatch(d)$ and $validate(p, M)$ are functions explained below.

The $generatePatch(d)$ function that takes the color sub-image d as input and generates a structure called patch. The patch contains a fixed-size image and the LBP histogram of that image.

```
$\boldsymbol{p\ =\ generatePatch(d)}$
1:  $d\ =\ rgb2gray(d)$
2:  $d\ =\ resize(d, \mathrm{T}, \mathrm{T})$
3:  $p.image\ =\ d - mean(d)$
4:  $p.lbp\ =\ LBPH^{u2}_{(8,2)}(p.image)$
```

Algorithm 3 - Generate Patch algorithm

We modified and extended the libfacerec[1] library to compute the $LBP^{u2}_{(8,1)}$, as described in section 4.2

The patch image $p.image$ is zero-mean normalized first to make it invariant to local illumination changes which is useful for computing the NCC later, not for the LBP itself. We resize the patch to account for variance in different scale detections. Bi-cubic interpolation is used.

$validate(d, M)$ is a function that decides whether or not the patch p should be kept in the model M. If the image d is very similar to another one already in the model M, then the patch is discarded. The zero-mean normalized correlation coefficient (ZM-NCC) is used to measure the similarity between the patches. The ZM-NCC at a point (u, v) for an image I and a template T is defined as:

[1] Libfacerec: https://github.com/bytefish/libfacerec

$$ZM\_NCC_{I,T}(u,v) = \frac{\sum_{x,y}[\,I(u+x,v+y)-\bar{I}\,][T(x,y)-\bar{T}]}{\sqrt{\sum_{x,y}[\,I(u+x,v+y)-\bar{I}\,]^2\sum_{x,y}[T(x,y)-\bar{T}]^2}} \quad (23)$$

$\bar{I},\bar{T}$ are the mean values for the image window being examined and the template respectively.

The zero-mean normalized correlation coefficient has the following properties:

1. $-1 \leq \mathrm{ZM}\_NCC \leq 1$
2. It is invariant to the average gray levels of the image window and the pattern

In our case we have two image patches p, q of the same size and we compute the $NCC_{p,q}$ at the center of the patches and $0 \leq x \leq patchWidth$, $0 \leq y \leq patchHeight$. The patches are already zero-mean normalized as shown in Algorithm 3, so during the computation we don't subtract their mean.

6.4 Face database and recognition

The face database is composed by several files saved on the hard drive. Those files include the face models of each person, the SVM model, and the file with the labels of each person. Each face model, as already discussed, is a collection of face patches (images) of size 50 x 50 pixels and their corresponding LBP features. A file containing all the patches from all the faces with their corresponding label (for each person) is also saved and used to train the SVM model of the classifier.

The recognition application loads the database on startup and then can predict the class (label) where a query face belongs to. Three types of classifiers are implemented: 1-NN, 3-NN and SVM, and the one to be used is selected in the configuration file. To use the SVM classifier, the SVM model and range files should be available. More information on how to re-train the SVM classifier and generate the model and range files, if required, is given in the Appendix C.

6.5 Recording a synchronized sequence

We modified the recording tool that was developed during the previous semester and enabled it to record hours of synchronized video from at least 6 network cameras. The output video of each camera has the original camera resolution size (1032 x 778) saved at 30 fps and encoded in MPEG-4 .avi format with a bit-rate of 51385 kb/s.

To allow concurrent smooth video presentation on the screen and recording, different threads have to be started that handle the saving. For this purpose, the class *videoRecorder* was built. It's basically a wrapper around the OpenCV's *VideoWriter* class. For each camera, one instance of *videoRecorder* is created and assigned to a new thread. This class reads from the corresponding camera buffer, passes the frame to the encoder and saves it to the specified output video.

7. CONCLUSIONS

We have presented a framework for real-time tracking and identification of people in a multi-camera setup. A face detector integrated with a people tracker for higher accuracy and faster detections is proposed. Models of the face, based on Local Binary Pattern Histograms, of various people are captured offline and stored to a database. An SVM classifier is built from those models and is able to recognize query faces with great variation in facial expressions, pose and illumination. The method is integrated with two trackers, one being the state-of-the-art Multi-Commodity Network Flow tracker [32] and is shown to increase its performance.

The method seems promising and it proves that other appearance information, except color models that is the most widely exploited appearance cue, can be used to boost the performance of trackers and generate identity-preserving trajectories. The fact that the face recognitions are proved reliable is very important since in many scenarios having the actual identity (and not only a label) of a person being tracked is critical. Also, the ability to distinguish trajectories of people that are stored in the database and "guests" can allow for new application scenarios.

7.1 Future Work

The work of this thesis can be extended in many ways. Some possible directions are the following:

- For people not present in the database, face models can be built on-the-fly to differentiate between individuals thus improving the trajectories.
- Other kinds of appearance cues can be incorporated in a similar manner, such as the height of a person or other soft biometrics. That information can't provide an identity to the person but still can minimize identity-switches by differentiating the people in the scene.
- Other face recognition techniques, more suitable for recognition on video sequences, can be explored. Also, a post-processing step can be added on the face modelling algorithm to remove redundant patches and provide faster recognitions.
- The spatial configuration of the cameras can be exploited. If, for a POM detection, a frontal face is detected in one camera, there is no need to search for a face for the same detection in another camera that sees the back of the head of the person. Also since multiple patches of each person's face are available during modelling, a 3D face model can be constructed, that will probably give better recognition results or allow for better analysis of facial expression (see next point).
- Once the face is detected in a scene, analysis of the facial expression, head pose estimation, eye location extraction or even gaze estimation can be performed (given sufficiently large video frames) for applications that this would make sense to be combined with tracking and/or identification. For example, tracking only the customers in a shop and locating the areas where they spend more time looking.

ACKNOWLEDGEMENTS

This thesis is the product of the work performed during my last semester at EPFL and it was based on the project carried out during the previous semester. Over this year spent at the Computer Vision Lab, many people helped me in different ways and I would like to thank all of them.

Most importantly, I’d like to thank my professor Pascal Fua, for supervising me during this thesis and giving me the opportunity to work in the laboratory and utilize all the resources. Also, I’d like to thank the second member of my committee, Mr. Richard Lengagne, for accepting to evaluate this work and. I am very grateful to Raphael Sznitman for giving me the code to incorporate the deformable detector to our system. I owe a special thanks to Horesh Ben Shitrit for his valuable feedback, the advice and all the help he provided during my master thesis. Finally, I’d like to express my gratitude to everyone who voluntarily participated in the recording of the video sequences used for this thesis.

The completion of this thesis was part of the graduate program that was co-funded through the Act “I.K.Y Bursary Program process with individualized assessment, academic year 2011-2012” by the O.P. resources "Education and Lifelong Learning" of the European Social Fund (ESF) and the NSRF 2007-2013.

A. VIOLA – JONES METHOD

Haar-like features

The proposed algorithm uses features instead of the direct value of the pixels. The features employed by the framework resemble the Haar wavelets, but they rely on more than one rectangular areas. The authors use four different types of features as shown in Figure 36. The value of the feature is computed as the difference between the sum of the pixels in the shadowed areas and the sum of the pixels within the clear areas. Those rectangular filters provide a coarser representation of the image compared to some alternatives, such as the steerable filters, but they are proved to capture the face characteristics and can be computed extremely fast at any scale and location.

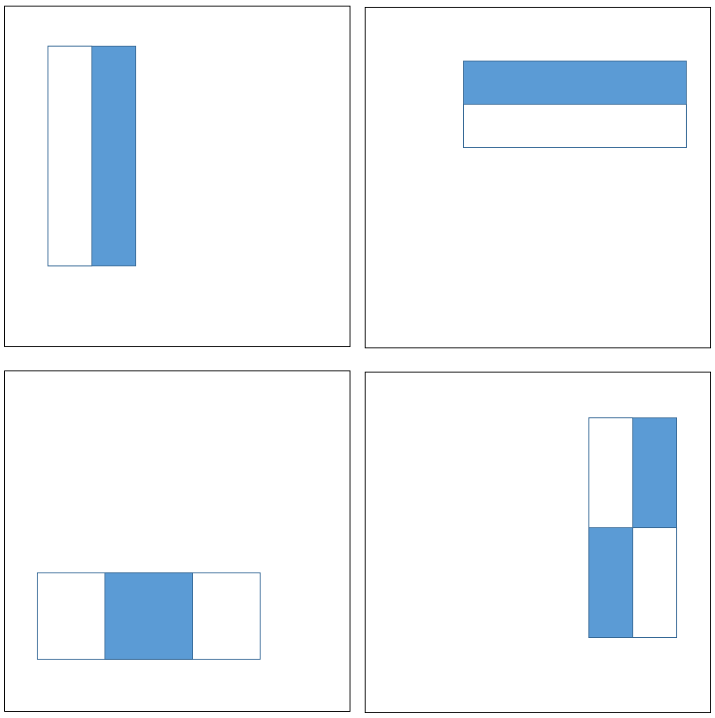


Figure 36 - The four types of Haar-like features used in the Viola - Jones method drawn inside a search window.

As is obvious, the set of all possible features of those four types that can be generated on an image window is extremely big; computing all of them would be computationally intensive and would produce redundant information. As explained on the next section, only a small subset of the exhaustive set of features is used.

One such feature of an image I can be efficiently computed in constant time on the integral image I' of I. The integral image is defined as follows:

$$I'(x,y) = \sum_{x' \leq x, y' \leq y} I(x',y') \qquad (24)$$

Which means that the value of the integral image at a point p is the sum of all the values of all pixels above and on the left of p (inclusive). The integral image can be computed in $O(N)$, where N is the number of pixels in the initial image.

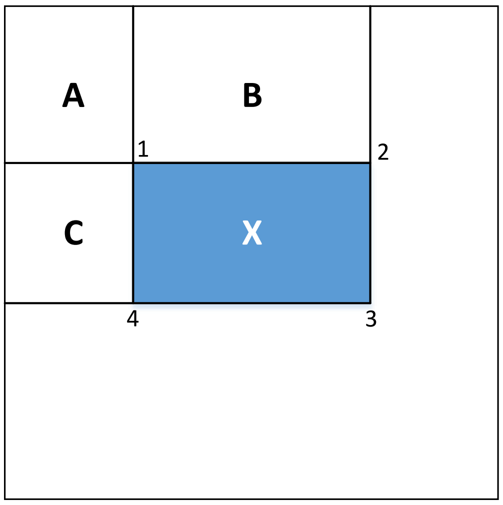


Figure 37 - Integral image. The sum of pixels in the rectangle X can be computed in O(1) by 4 operations.

Figure 37 shows an integral image I'. The sum of the pixels within the rectangle X can be efficiently computed in $O(1)$ with just four array references. The value at position 1 is the sum of pixels in rectangle A, $I'(1) = A$. Respectively, $I'(2) = A + B,\ I'(3) = A + C$ and $I'(4) = A + B + C + X$. Therefore:

$$X = I'(4) + I'(1) - (\ I'(2) + I'(3)\ ) \tag{25}$$

Classifier

As already discussed, there is a very large number of rectangle features that can be computed for each sub-window of the image and even though it is extremely fast to compute a feature, the computation of all these features would be too expensive. So the authors propose a method to find a small number of features that combined can create an effective classifier.

A modified AdaBoost [36] method is employed to find the set of features, as well as to train the classifier. Each weak classifier depends on a single feature and at each round of the boosting, the method selects the optimal feature that best separates the positive and the negative (weighted) examples. A weak classifier is defined as:

$$h(x, f, p, \theta) = \begin{cases} 1, & pf(x) < p\theta \\ 0, & otherwise \end{cases} \tag{26}$$

Where x is the variance-normalized image patch (24 x 24 pixels), f is the feature, θ is the threshold and p is the polarity. So at each step, both the feature and the threshold are determined. As per AdaBoost method, after the selection the weights of the examples are updated so that misclassified examples have a larger weight on the subsequent step. That way the next selected feature will try to best separate the previously misclassified examples. In the end, after T steps of the AdaBoost method, a strong classifier $C(x)$ is built as a linear combination of the T weak classifiers.

$$C(x) = \begin{cases} 1, & if \quad \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ 0, & \qquad\qquad otherwise \end{cases} \tag{27}$$

The boosting algorithm is shown below:

```
1: Input: Training set $\{(x_i, y_i), \dots, (x_n, y_n)\}$, where $x_i$ is an image and $y_i = 0, 1$ for negative and
   positive examples respectively
2: If $m, l$ the number of negative and positive examples respectively, initialize $w_{1,i} = \frac{1}{2m}$ for the
   negative examples and $w_{1,i} = \frac{1}{2l}$ for the positive examples
3: For $t = 1, \dots, T$:
      a. Normalize: $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
      b. Select the best weak classifier with respect to the weighted error:
         $$\varepsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$
      c. Set $h_t(x) = h(x, f_t, p_t, \theta_t)$ where $f_t, p_t, \theta_t$ are the minimizers of $\varepsilon_t$.
      d. Update the weights: $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$ , where $e_i = 0$ if $x_i$ is classified correctly, or
         $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
4: The strong classifier is: $C(x) = \begin{cases} 1, & if \quad \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ 0, & otherwise \end{cases}$
   where $a_t = \log \frac{1}{\beta_t}$
```

Algorithm 4 - Viola - Jones classifier boosting algorithm

For more information about the algorithm and the optimization step 3b of the algorithm, the reader is referred to [2].

An illustration of the AdaBoost method is shown in Figure 39. There are two classes of items (blue, orange). The size of the items denotes its weight. The items marked with **X** are the misclassified. On each step of the AdaBoost algorithm, the weights are updated so that the ones that were wrongly classified will have greater weight on the following iteration. The final strong classifier is a linear combination of the weak classifiers. For example, in the case where each weak classifier is a perceptron, the final strong classifier will be a two-layer perceptron.

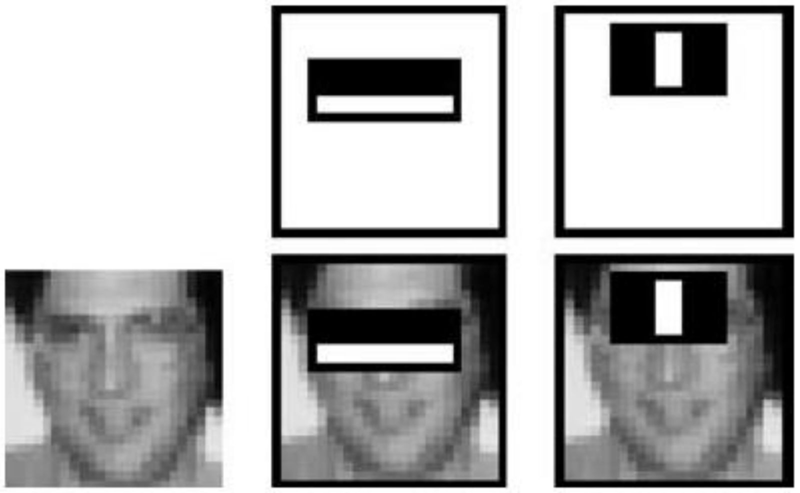
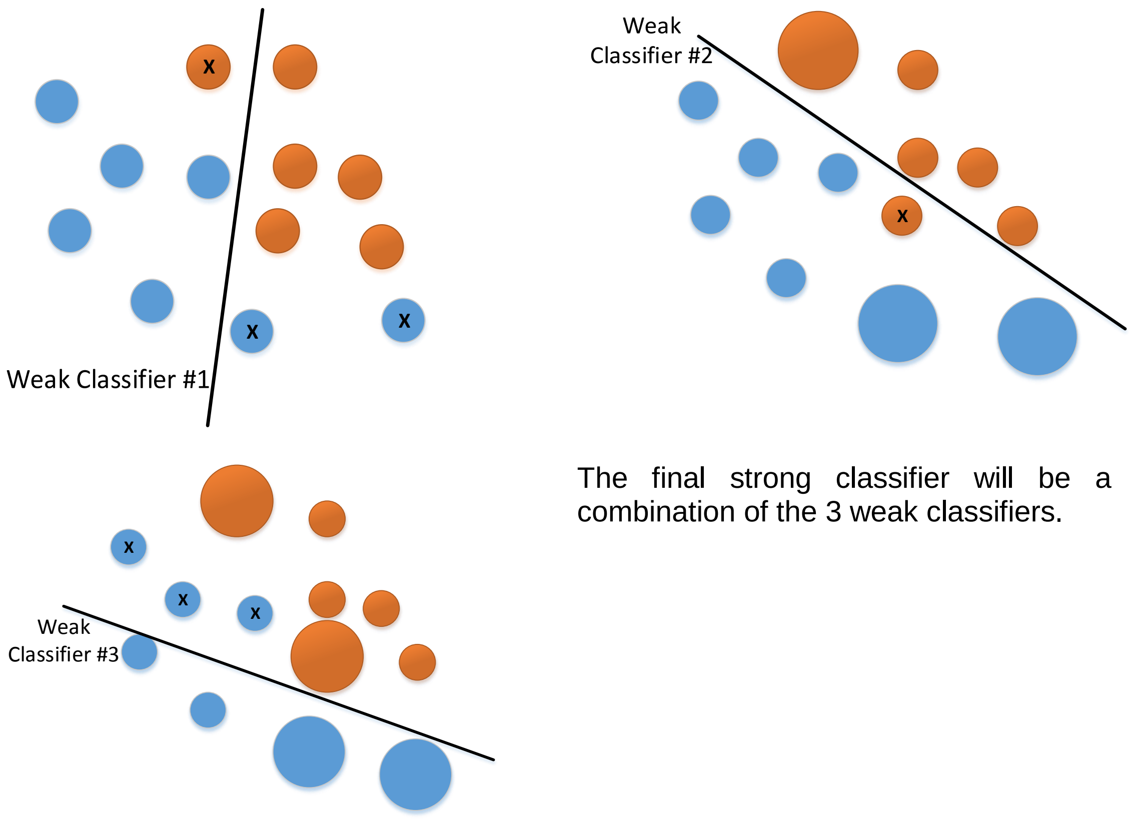


Figure 38 - The first two features of the Viola- Jones as selected by the boosting algorithm are shown in the top row. Below they are overlayed on top of the image patch. The first features captures the darker area in the region of the eyes compared to the area exactly below it. The second feature captures the brighter area in between the two eyes (image from [2]).



The final strong classifier will be a linear combination of the 3 weak classifiers.

Figure 39 – An illustration of 3 steps of the AdaBoost method. The orange and blue color represent the real class in which each instance belongs. The 'X' sign in a circle denotes that this instance was wrongly classified by the weak learner. The size of an instance denotes its weight. At each iteration, the instances that are wrongly classified, receive larger weights so that they can be correctly classified later by another weak classifier. A linear combination of the weak classifier creates the final strong classifier.

A classifier consisting of the first two features selected by the boosting method is shown in Figure 38. The first one (horizontal) captures the darker area of the eyes in contrast with whiter area below them. The vertical feature captures the whiter area between the two darker eyes. These two simple features reject about 50% of non-faces, while detecting almost 100% of the faces. The next ten features still detect almost 100% of the faces while rejecting about 80% of the non – faces. This drives the idea of the cascading classifiers as explained in the next chapter. We can very fast get rid of the majority of the image, before spending more computational power on areas where there might be a face.

Cascade of classifiers

As already discussed, a main contribution of the Viola – Jones work, was the method for combining successively more complex classifiers in a cascade style to increase the speed of the detector, by rejecting non-face regions of the image very quickly. A schematic representation of this approach can be seen below (Figure 40). The authors show that there is practically little difference in terms of accuracy between a monolithic classifier and a cascaded one. However, in terms of speed, the cascaded classifier is proved to be 10 times faster. The first classifier C_1 consists of the two features shown in Figure 38, while the second classifier C_2 contains 10 features and can reject about 80% of the non-face regions while maintaining almost 100% detections of faces.

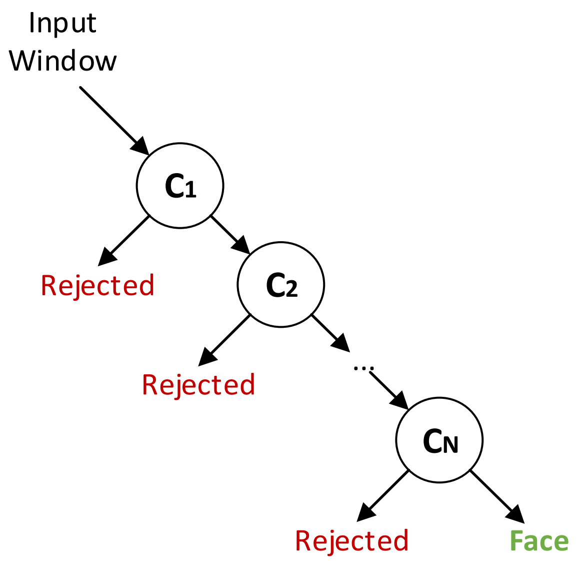


Figure 40 – Cascade of classifiers. If a patch is rejected by a classifier, the detection stops for this region and the patch is dropped. Classifiers of the upper layers have very few features and their response is computed very fast. They have the advantage that they can prune large area of the image where it's sure that there isn't a face.

The complete algorithm for building the cascade of classifiers can be found in [2].

Detecting faces in an image

To detect a new face in an image, a search window is swiped across the image with a predefined step size and it is evaluated against the cascaded classifier. Viola and Jones removed the need for an image pyramid, as was usually done to search for detections in different scales, by scaling the rectangle filters respectively. That way the detection was much faster, since only the integral image was required to be computed and then the features could be evaluated at any scale and location as described before. It is possible that there will be multiple detections for the same face as the search window is shifted along the image. To solve this problem, multiple detections are merged into one as the final step of the algorithm.

B. ASSIGNMENT PROBLEM

Let $\mathrm{G} = \{\mathrm{g}_1, \mathrm{g}_2, \dots, \mathrm{g}_\mathrm{N}\}$ and $D = \{d_1, d_2, \dots, d_N\}$ denote two sets of points in some space. Let $R_{N\times N}$ be a matrix containing some distance metric between every pair of $g \in G$ and $d \in D$ such that $R(i,j) = distance(g_i, d_j)$.

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,N} \\ r_{2,1} & & r_{2,N} \\ \vdots & \ddots & \vdots \\ r_{N,1} & \cdots & r_{N,N} \end{bmatrix} \tag{28}$$

The assignment problem requires to find a bijection (one-to-one correspondence) $f: G \to D$ such that the quantity below is minimized:

$$\sum_{i=1}^{N} R(i, f(i)) \tag{29}$$

This can be formulated as linear optimization program:

$$\begin{aligned} &\text{Minimize} && \sum_{i\in O}\sum_{j\in M} R(i,j)\cdot a_{ij} \\ &\text{Subject to} && \textstyle\sum_{D_i\in D} a_{ij} = 1 \;\; \forall G_j \in G \\ &&& \textstyle\sum_{G_j\in G} a_{ij} = 1 \;\; \forall D_i \in D \\ &&& a_{ij} \geq 0 \; \forall i \in G, j \in D \end{aligned} \tag{30}$$

This is a well-known optimization problem (the Assignment Problem) and is proved to have an optimal solution [47] where the variables a_{ij} take the value 1 if the assignment between G_i and D_j is done, or 0 otherwise. The problem can be solved in polynomial time using the Hungarian Method. The analysis of this algorithm is out of the scope of this report and the reader is referred to [47], [48] for more information. In the above, we assumed that $|G| = |D|$. This isn't always the case. To solve this problem we augment as many columns (or rows) as necessary so that both sets have equal value. All augmented elements have value ∞. After the optimization problem is solved, the assignments containing augmented elements can be dropped.

C. EXECUTING THE CODE

Face recognition / modelling pipeline execution example

A number of scripts and configuration files have been created to ease the generation of a video dataset and the execution of the various stages of the tracking pipeline. All folders are relative with the root directory being the directory where the code accompanying this thesis is extracted. It assumes that a "data/faceModels" and "data/faceExperiments" directories are also present containing the respective "TEMPLATE" directories (see accompanying code for more information) and the labels.txt file. For any sequence that you want to generate with the name **datasetName**, the proposed directory structure is the following (you can copy it from the "TEMPLATE" directory or use the createFaceModelDir / createExperimentDir scripts described later).

```
datasetName/
        videos/ - Folder containing the videos
        bgs_videos/ - Background subtraction videos (optional)
        bgs_images/ - Background subtraction images
        bg_model/ - Background model
        pom_prob/ - POM probabilities
        pom_synthetic/ - Synthetic images generated by POM (optional)
        calibration/ - The calibration files (optional)
        faces/ - Face detections and recognitions
        config-bg.xml – Configuration file for the background subtraction
        datasetName.pom – POM Configuration file
        datasetName.kal – Configuration file for Kalman tracking
        datasetName.mconf – Configuration file for detection and modelling
        or
        datasetName.rconf – Configuration file for detection and recognition
```

Step 1 (depending on the dataset)

Process the video sequences of a person to create a face model:

```
scripts/createFaceModelDir.sh <label> <personName>
```

where <personName> is a single word, <description> can be the full name or something else (make sure to double quote if it's more than one word) and <label> is a positive integer (that shouldn't already be present in the labels.txt file). This will create a folder structure similar to the above and will also edit the labels.txt file to include the new face entry. Make sure not to use a personName / label that already exists in the file.

If you created the folder structure manually (or if otherwise needed), make sure to edit the configuration files in the "data/faceModels/personName" directory.

Process videos for face recognition and tracking:

If we need to create a video sequence in which we want to recognize the faces of the people, execute the script:

```
scripts/createFaceModelDir.sh <datasetName>
```

This will create the required directory structure under the folder data/faceExperiments/datasetName. The configuration file for the recognition application will be named *datasetName.rconf*.

Step 2

To save the video from each camera, the background model and the scene data, run the save_video_model_high application (see 6.5).

```
code/recording_tool/save_video_model_high
```

Step 3

The complete pipeline can be executed using the script:

```
scripts/pipeline.sh <dataset_type> <dataset_name>
```

where <dataset_type> is either faceModels or faceExperiments. The individual steps that are executed by this scrip are presented below.

Executing individual applications

Background subtraction

```
scripts/bgsub.sh <datasetFolder>
```

The data/ folder is always prefixed to the “datasetFolder”. If the dataset root directory is “data/face_models/george/” then execute the following:

```
scripts/bgsub.sh faceModels/george
```

Open Source POM software

```
packages/pom/pom <path to .pom config file>
```

Face detection without POM (optional)

To run the face detection software (version without POM), do the following for each camera / video (the .fdc will need editing, or use the command line parameters of the application):

```
code/face_detection/face_detection <path to .mconf file>
```

Face detection with POM

To run the face detection software, using POM detections:

```
code/face_detection/face_detection <path to .mconf file>
```

Face modelling

```
code/face_modelling/face_modelling <path to .mconf file>
```

This will generate the face model .lbpm file in the dataset folder.

Face detection and modelling (in one application – preferred solution)

```
code/modelling_pipeline/modelling_pipeline <path to .mconf file>
```

Face recognition

```
code/face_recognition/face_recognition <path to .rconf file>
```

Face detection and recognition (in one application – preferred solution)

```
code/recognition_pipeline/recognition_pipeline <path to .rconf file>
```

Kalman tracker

Using both color model and face recognitions:

```
code/tracking_kalman/tracking_color_faces <path to .kal file>
```

Using only face recognitions:

```
code/tracking_kalman/tracking_faces <path to .kal file>
```

Using only color model:

```
code/tracking_kalman/tracking_color <path to .kal file>
```

Kalman tracker without appearance cues:

```
code/tracking_kalman/tracking <path to .kal file>
```

Re-training the SVM classifier

To train the SVM classifier all that is required is to generate a file with all the LBP features from every person in the database, by calling the GenerateLBPDataFile() of the FaceDatabase class, and then executing the script:

```
packages/libsvm/tools/easy.py <the lbp file>
```

This will generate the SVM model file and the SVM range file that can be loaded by the FaceDatabase to support the SVM classification.

Enabling support for more cameras

The configuration tool of the framework was extended to support an arbitrary number of Ethernet cameras. When a new camera is connected to an Ethernet adapter of the computer, the Reverse Path Filtering of the kernel for that adapter should be turned off, for the driver to be able to communicate with the camera. To do this, run the command:

sysctl -w net.ipv4.conf.ethX.rp_filter=0

where X is the adapter number where the camera is connected. This change is temporary until the system is rebooted. To make the change permanent edit the file */etc/sysctl.conf* and set the same variable as in the command.

REFERENCES

[1] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, no. 9, p. 1820–1833, 2011.

[2] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.

[3] Y. Abramson, B. Steux and H. Ghorayeb, "YEF real-time object detection," in *International Workshop on Automatic Learning and Real-Time (ALaRT)*, Siegen, 2005.

[4] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *International Conference on Image Processing (ICIP)*, Rochester, 2002.

[5] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," Microsoft Research, Redmond, 2010.

[6] K. Levi and Y. Weiss, "Learning object detection from a small number of examples: the importance of good features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, 2004.

[7] R. Feraund, O. Bernier, J. Viallet and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 1, no. 23, pp. 42-53, 2001.

[8] P. Wang and Q. Ji, "Multi-view face detection under complex scene based on combined svms," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, Cambridge, 2004.

[9] K. Ali, F. Fleuret, D. Hasler and P. Fua, "A real-time deformable detector," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 2, pp. 225-239, 2012.

[10] Z. Kalal, K. Mikolajczyk and J. Matas, "Face-tld: Tracking-learning-detection applied to faces," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Hong Kong, 2010.

[11] Z. Kalal, K. Mikolajczyk and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 7, pp. 1409-1422, 2012.

[12] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion (CVPR)*, Providence, 2012.

[13] M. Yang, D. Kriegman and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, p. 34–58, 2002.

[14] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.

[15] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 12, no. 1, pp. 103-108, 1990.

[16] D. Swets and J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 18, no. 8, pp. 831-836, 1996.

[17] P. Belhumeur, J. Hespanha and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 711-720, 1997.

[18] A. Martinez and A. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 2, pp. 228-233, 2001.

[19] T. Ahonen, A. Hadid and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, no. 12, pp. 2037-2041, 2006.

[20] A. Abate, M. Nappi, D. Riccio and G. Sabatino, "2D and 3D face recognition: A survey," *Pattern Recognition Letters,* vol. 28, no. 14, oct 2007.

[21] X. Liu, T. Chen and S. Thornton, "Eigenspace updating for non-stationary process and its application to face recognition," *Pattern Recognition, Special issue on Kernel and Subspace Methods for Computer Vision,* vol. 36, no. 9, pp. 1945-1959, 2002.

[22] H. Wang, Y. Wang and Y. Cao, "Video-based face recognition: A survey," *World Academy of Science, Engineering and Technology,* vol. 60, pp. 293-302, 2009.

[23] C. Chan, "Multiscale Local Phase Quantisation for Robust Component-based Face Recognition using Kernel Fusion of Multiple Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 99, no. 1, 2012.

[24] R. Jafri and H. Arabnia, "A survey of face recognition techniques," *Journal of Information Processing Systems,* vol. 5, 2009.

[25] J. Black and T. Ellis, "Multi camera image tracking," in *International Workshop on Performance Evaluation of Tracking and Surveillance*, Kauai, 2001.

[26] A. Mittal and L. Davis, "M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene," *International Journal of Computer Vision (IJCV),* vol. 51, no. 3, pp. 189-203, 2003.

[27] Z. Khan, T. Balch and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 27, no. 11, pp. 1805-1819, 2005.

[28] K. Okuma, A. Taleghani, N. Freitas, J. Little and D. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *European Conference on Computer Vision (ECCV)*, Prague, 2004.

[29] V. Singh, B. Wu and R. Nevatia, "Pedestrian tracking by associating tracklets using detection residuals," in *IEEE Workshop on Motion and video Computing (WMVC)*, 2008.

[30] J. Henriques, R. Caseiro and J. Batista, "Globally optimal solution to multi-object tracking with merged measurements," in *IEEE International Conference on Computer Vision (ICCV)*, Barcelona, 2011.

[31] J. Berclaz, F. Fleuret, E. Turetken and P. Fua, "Multiple Object Tracking Using K-Shortest Paths Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, no. 9, pp. 1806 -1819, sep 2011.

[32] H. Ben Shitrit, J. Berclaz, F. Fleuret and P. Fua, "Multi-Commodity Network Flow for Tracking Multiple People," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2013.

[33] R. Mandeljc, S. Kovavcivc, M. Kristan and J. e. a. Pervs, "Tracking by Identification Using Computer Vision and Radio," *Sensors,* vol. 13, no. 1, pp. 241-273, 2012.

[34] F. Fleuret, J. Berclaz, R. Lengagne and P. Fua, "Multicamera People Tracking with a Probabilistic Occupancy Map," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, no. 2, pp. 267-282, feb 2008.

[35] A. Cohen and V. Pavlovic, "An efficient IP approach to constrained multiple face tracking and recognition," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Barcelona, 2011.

[36] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[37] F. Fleuret and D. Geman, "Stationary features and cat detection," *Journal of Machine Learning Research,* vol. 9, pp. 2549-2578, 2008.

[38] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *EURASIP Journal on Image and Video Processing,* vol. 2008, no. 1, May 2008.

[39] H. Rowley, S. Baluja and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 1, pp. 23-28, 1998.

[40] T. Ojala, M. Pietikainen and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition,* vol. 29, no. 1, pp. 51-59, 1996.

[41] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 971-987, 2002.

[42] C. Shan, S. Gong and P. McOwan, "Facial expression recognition based on Local Binary Patterns: A comprehensive study," *Image and Vision Computing,* vol. 27, no. 6, pp. 803-816, 2009.

[43] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 3, p. 27:1–27:27, 2011.

[44] M. Zervos, "Real-time multi-object tracking using multiple cameras," Semester Project Report, EPFL, Lausanne, 2012.

[45] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering,* vol. 82, pp. 35 - 45, 1960.

[46] H. Ben Shitrit, J. Berclaz, F. Fleuret and P. Fua, "Tracking multiple people under global appearance constraints," in *IEEE International Conference onComputer Vision (ICCV)*, Barcelona, 2011.

[47] R. Burkard, M. Dell'Amico and S. Martello, Assignment Problems, Society for Industrial and Applied Mathematics, 2009.

[48] H. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly,* vol. 2, no. 1-2, pp. 83-97, 1955.