J Grid Computing
https://doi.org/10.1007/s10723-018-9467-x

Distributed Video Surveillance Using Smart Cameras

Hanna Kavalionak · Claudio Gennaro · Giuseppe Amato · Claudio Vairo · Costantino Perciante · Carlo Meghini · Fabrizio Falchi

Received: 28 November 2017 / Accepted: 19 September 2018

Abstract Video surveillance systems have become an indispensable tool for the security and organization of public and private areas. Most of the current commercial video surveillance systems rely on a classical client/server architecture to perform face and object recognition. In order to support the more complex and advanced video surveillance systems proposed in the last years, companies are required to invest resources in order to maintain the servers dedicated to the recognition tasks. In this work, we propose a novel distributed protocol for a face recognition system that exploits the computational capabilities of the surveillance devices (i.e. cameras) to perform the recognition of the person. The cameras fall back to a centralized server if their hardware capabilities are not enough to perform the recognition. In order to evaluate the proposed algorithm we simulate and test the 1NN and weighted kNN classification algorithms via extensive experiments on a freely available dataset. As a prototype of surveillance devices we have considered Raspberry PI entities. By means of simulations, we show that our algorithm is able to reduce up to 50% of the load from the server with no negative impact on the quality of the surveillance service.

Keywords Distributed architectures · Internet of things · Video surveillance · Self-organization

H. Kavalionak (✉)
University of Florence, via Morgagni 65, Firenze, Italy
e-mail: hanna.kavalionak@unifi.it

C. Gennaro · G. Amato · C. Vairo · C. Perciante · C. Meghini · F. Falchi
Information Science and Technologies Institute, National Research Council (ISTI-CNR), Pisa, Italy

C. Gennaro
e-mail: claudio.gennaro@isti.cnr.it

G. Amato
e-mail: giuseppe.amato@isti.cnr.it

C. Vairo
e-mail: claudio.vairo@isti.cnr.it

C. Perciante
e-mail: costantino.perciante@isti.cnr.it

C. Meghini
e-mail: carlo.meghini@isti.cnr.it

F. Falchi
e-mail: fabrizio.falchi@isti.cnr.it

1 Introduction

Video surveillance is of paramount importance in areas like law enforcement, military and even for commercial environment. A way to execute the surveillance is to stream the data from the cameras to the displays of the human operators, who are responsible to analyze the video. Human resources used in the field of the video surveillance services are both costly and not reliable. The person who is supposed to follow and analyze the surveillance video cannot keep the concentration for a long time, and can miss some important information on the video. Hence, with the recent progresses in smart technologies, automated video surveillance, where the video streams are analyzed automatically, as you can see in the following surveys [30, 35], gained a lot of interest.

Automated area surveillance addresses the real time observation of people and objects in a busy environment, leading to their recognition and description of their actions and interactions. Being one of the most important technique to organize a secure area, the surveillance smart service problems became an hot research issue. The most important issues include tasks like people detection, recognition and tracking; object recognition, motion analysis, etc [38, 42].

The motivating scenario that we consider for this work is that of face recognition. The surveillance system should be able to recognize faces, and track their movements in possibly large area (e.g. an airport, a city district, a campus, etc.), using various cameras appositively installed. Executing the needed image processing tasks, for offering this kind of automated video surveillance in real-time, requires significant storage and processing resources. The need to process big amount of video data represents a common bottleneck of the automated video surveillance systems, especially when the number of camera providing video streams and the number of faces to be simultaneously recognized and tracked is large.

The aim of this work is to analyze the issues and solutions related to resource allocation, for executing automated video surveillance, in a fully distributed environment [4, 5]. In particular, we suppose that the (smart) camera devices themselves can cooperate to execute the needed faces detection, recognition and visual analysis tasks. In this work, we study how the image recognition algorithms can be orchestrated across several devices so that bottlenecks are reduced and resource can be managed more effectively.

One of the straightforward approaches for resource allocation in video surveillance services is to use client-server model of communication. In other words, the surveillance devices stream the video directly to a main powerful server, where the data can be analyzed [23]. Despite of the effectiveness and simplicity of such architecture, this approach presents several negative sides. First, relying all the functionality on a single server creates a bottleneck for the system security and reliability. A high rate of the recognition tasks on the server can lead to delays in the image processing. Second, in order to analyze all the video streams coming from the surveillance devices the target organization needs to maintain big and costly infrastructure of servers dedicated only to the surveillance task. Moreover, such a service become too costly for small and medium size business, which are not able to allocated so much financial resources only for the surveillance.

This work extends and enhances our previously published research work [16], where we have proposed a distributed algorithm for load balancing between Smart Sensing Units for video surveillance task. The adaptive algorithm distributes, at run time, the recognition tasks between the resources of surveillance devices and servers. The detection and recognition tasks are executed locally by surveillance devices, which exploit both the spatial and temporal topology of the moving people, to cache and reuse locally parts of the classification features. Only when devices are not able to execute the recognition task with the cache, a recognition request is sent to the server. We extend the previously proposed adaptive algorithm considering the overhead of real classification techniques. In our evaluation we implemented and tested classification algorithms for face recognition: 1NN and the weighted kNN classifiers with Local Binary Pattern (LBP)[7].

The paper is structured as follows. Section 2 briefly describes the background for the classification algorithms we considered in our work. Section 3 provides the definition of the system model and problem statement, whereas Section 4 presents the algorithm for the adaptive camera-assisted person recognition. Section 5 evaluates the classifiers in terms of classification performance and accuracy. Section 6 evaluates the distributed surveillance solution through simulations and Section 7 offers an overview of the related work. Finally, we conclude the paper in Section 8 by discussing the results and the future work.

2 Background

Face identification in images, or frames, coming by a video source consists of a two steps pipeline: face detection and face recognition.

Face detection allows a computer to identify human faces in digital images and video. This task is not trivial due to the fact that human face in video frames is usually a dynamic object, with high degree of variability in its appearance. Different illumination conditions, facial poses and expressions, occlusion, rotation can affect a face detection algorithm. The problem of face detection is well presented in the literature and for a comprehensive survey we refer the reader to [41].

Face recognition problem today is still a hot research topics. It is a classification problem in which a new detected face must be matched against an available dataset of already labelled faces. In order to efficiently represent those faces, algorithms are used to extract facial features. Among these, the Eigenface method based on PCA [32], the Fisherface method based on LDA [10] and Elastic Bunch Graph Matching (EBGM) that approximates a face through a deformable graph [37] have obtained success and are largely used in face recognition applications. More sophisticated features based on Artificial Intelligence and its new research fields, such as Deep Learning [31], have been proposed too. These algorithms achieve great accuracy but are very computationally demanding. Thus, in order to meet the requirements of the real-time applications and of the embedded computing boards, simple algorithms are required for fast feature extraction. Among these, the LBPH [2] has been investigated and is used in the system since it better fits the requirement of a dynamic system in which classifiers can be added with respect to the Fisherfaces model. The aim of the facial features seen so far is to let the system to discriminate people. However, a convenient way to assess, with a certain degree of confidence, if a new unlabelled face belongs to someone that system already knows is to adopt a classification approach. This is a recognition task that can be formulated as follows: given a set of labels or classes $\{c_1, c_2, c_3, \ldots, c_h\}$, a set of already labelled objects $\{o_1, o_2, o_3, \ldots, o_n\}$ (belonging to the *training set*, TS) and a query q (i.e., an unlabelled object), the problem is to assign a label to q.

In this paper, we consider the following classifiers for face recognition: the first nearest neighbor (*1NN*) and the weighted k-nearest-neighbor (*weighted kNN*) implemented with LBP.

1NN Classifier Given an unknown face q and a set of known and labelled faces (o_i, c_i), the *1NN* algorithm recognizes q according to the label of the nearest object belonging to the training set for the query, in the feature space. The confidence for the response of the *1NN* classifier is evaluated as

$$c_{1NN} = 1 - \frac{d(q, o_j)}{d_{max}} \tag{1}$$

where o_j denotes the nearest face to the query q, $d(q, o_j)$ is the distance of the facial faces q and o_j, and d_{max} is a distance used as normalization factor. In case of ties, different approaches can be followed such as peeking at random a label among the winners. The aim of the algorithms for facial features extraction is to generate patterns that lie near each other for the same faces and are far away from the features of different faces.

Weighted kNN Classifier The key idea of the $weighted\ kNN$ is to assign a *weight* or *score* s_{o_i} to an object o_i, according to its position with respect to the query. This score is assigned to each object belonging to the result set that the kNN would return and is evaluated as $s_{o_i} = 1 - \frac{d(q,o_i)}{d_{max}}$. Then, the algorithm evaluates the sum of this scores $z(q, c_j)$ for the different classes c_j present in the result set. In the end, the label assigned to the query will be the one of the class that reaches the maximum value. In this case, the confidence value returned by the algorithm has the following form:

$$c_{wkNN} = 1 - \frac{z(q, c_{w_2})}{z(q, c_{w_1})}, \tag{2}$$

where c_{w_i} denotes the class that obtained the i-th higher score in $kNN(q)$. This recognition confidence can be used to decide whether or not the predicted label has a high probability to be correct. Another kind of confidence we analyze is the following:

$$c_{wkNN_2} = \frac{\sum_i s_{o_i}}{k} \quad \forall o_i \in c_{w_1} \tag{3}$$

This value expresses the probability that q belongs to c_{w_1}. The term k at the denominator is needed in order to have a value ranging in [0, 1]. The weighting schema reduces the probability of ties.

3 System Model and Problem Statement

In our work, we model the geographical area, where the surveillance system is deployed, as divided into sectors. A set of *Smart Sensing Units* (SSUs) are distributed among the sectors in order to execute the

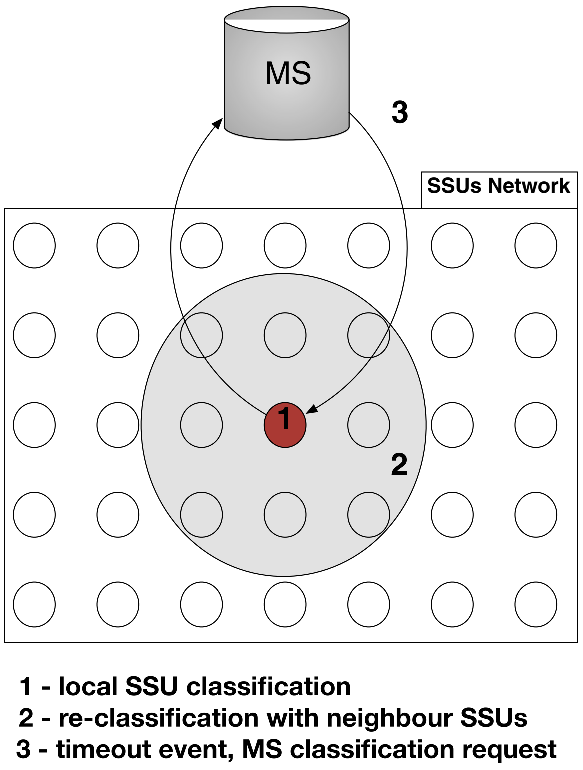

MS
3
SSUs Network
1
2
1 - local SSU classification
2 - re-classification with neighbour SSUs
3 - timeout event, MS classification request

Fig. 1 System model

video surveillance task (Fig. 1). Such units have various sensing, computing, storage, and communication capabilities. It is important to notice that a SSU is a logical unit, which does not necessarily correspond to one single physical sensor. For instance, a SSU unit can be a smart camera having on board sensing, computing, storage resources. However, it can also be composed of a camera connected to a computer. Moreover, we assume also the existence of a *Main Server* (MS) unit in the network. A main server is a stand alone SSU that is characterized by (theoretically) infinite computing, storage, and communication capabilities. It also plays the role of the central server where the devices send their requests, in case of client-server modality.

In our work, we distinguish two possible scenarios: *basic* and *enhanced*. In the basic scenario, all SSUs are connected to the central MS. Moreover, the functionality of SSUs is limited to the video observation and data streaming to the MS. In the enhanced scenario, on the other hand, all SSUs are connected in a structured peer-to-peer topology. SSUs can cooperate with each other in order to execute complex tasks and build together a rich representation of the context where the infrastructure is deployed. We consider the peer-to-peer overlay connecting the SSUs to be fixed. Moreover each of SSUs has the possibility to contact directly the MS and its geographical neighbor SSUs. We assume the communication between SSUs is reliable, i.e., no message is ever lost or corrupted. Moreover, in the enhanced scenario each SSU has computational resources and local storage memory, where it keeps a *recognition library*. The recognition library is a collection of recognizers, each specialized to recognize a certain person. The MS's recognition library contains the recognizers for all the recognizable persons in the surveillance area, while the recognition library of an SSU is basically a local cache that keeps only a subset of all recognizable persons.

As an illustrative example, let us consider the following scenario, taking place in a geographical area (the surveilled area) of appropriate size, where a number of SSUs have been deployed for video surveillance purposes. Here we consider a person that is moving in the area. The system should visually recognize the person in order to evaluate the presence of this person in the area. One of the straightforward solutions for area video surveillance is to stream all the data from the SSUs to the MS for the following processing and storage. This approach can cause some additional system issues. At first, the interested organization needs to have a powerful server system devoted specially for the surveillance needs. This server system needs to be powerful enough to continuously process the video streaming data that is coming from the SSUs. The second problem is connected with the delays for the person recognition, caused by possible server overloading and network bandwidth overhead in case of a high rate of requests for recognition.

In order to tackle these issues, our solution exploits a distributed architecture for face recognition, where the detection and recognition tasks are moved to SSUs when it is possible. We extensively test SSUs in a wide variety of scenarios with the objective to perform the most possible computation on the camera themselves and using optimized strategies of distributed computation to solve the most challenging resource demanding tasks. Moreover, we also implemented three classification algorithms and extensively tested them on a public available dataset for different scenarios.

4 Algorithm

In this section, we describe our enhanced algorithm for distributing the recognition processes on the SSUs. Each camera in the system keeps the list of its neighbors SSUs, which we name $view$. In order to improve the efficiency of the recognition process each camera in the area has associated computational resources and can locally keep a recognition library that contains a subset of all recognizers for *faces*. Using a subset of these local recognizers each camera in the network tries to recognize the face of the detected person locally and with a help of neighbor SSUs, without involving the MS. The rationale behind it is that we reduce the computational and bandwidth consumption impact to the MS. Each entry in the recognition library is composed of (1) a set of classifiers for a face, *class* in the following, (2) the alarm status (*alarm* or *recognized*) and (3) a counter *age* that indicates the number of active thread cycles passed from the last time when the corresponding face has been seen by the SSU. The size of the library is limited to *recognizersLimit* according to the available storage space of a SSU.

Three threads are executed by each of the SSUs: (1)*active thread* (Algorithm 1) is responsible for the active area monitoring, (2)*time-out thread* corresponds to the timeout event in the face recognition by neighbors and (3) *passive thread* (Algorithm 2) processes the incoming messages received by an entity.

Every ΔT, each camera executes the active thread Algorithm 1. At first, it increases the "age" of all known recognizers (increaseAge()) in the library *recognizers*. In the next step, a camera processes the current surveillance area image and extracts the facial features getVisibleFaces() for the detected persons. During each cycle a SSU selects a subset *selection* of *recognizers* from the local collection (selectRecognizers). The SSU selects $n_{selection}$ of *recognizers* with the lowest *age* parameter. The choice of the last viewed face classes allows us to exploit the spatial and temporal locality of the moving people in the face recognition. The size of the *selection* subset is computed according to the equation:

$$n_{selection} = \frac{T_{max} - T_{ms} - T_{recl} - t_{det}}{ct \cdot n_{events}} \qquad (4)$$

where T_{max} is a predefined maximum allowed face recognition delay, T_{ms} is the estimated average time

Algorithm 1 Active thread algorithm executed by SSU

```
repeat
    recognizers.increaseAge();
    events ← getVisibleFaces();
    selection ← selectRecognizers(recognizers);
    foreach face in events do
        if alarm ← getStatus(face, selection) then
            send(FACEID, face, alarm, view);
            startAlarmActivity();
            return;
        if recognized ← getStatus(face, selection) then
            send(FACEID, face, recognized, view);
            return;
        if null ← getStatus(face, selection) then
            $t_\delta$ ← setRecognitionTimeout();
            reclBuffer.add(face, $t_\delta$);
            send(ALARMREQUEST, face, null, view);
            return;
    wait $\Delta T$;
until;
```

needed for the MS recognition, T_{recl} is the estimated average time period required for the recognition by the neighbor SSUs, t_{det} is the estimated time of the features extraction, ct is the time needed to compare the extracted features with one class in the recognition library and n_{events} is the number of detected faces in the surveillance area.

For each *face* in *events* a SSU executes the recognition algorithm. In case *face* corresponds to the entry in *recognizers* with *alarm* status, the SSU (1) sends a notification message type FACEID with an *face* features and the *alarm* tag to the neighbor SSUs; then (2) starts predefined alarm activity, for example video recording and live streaming to the operator displays. The precise activity to be executed after the triggering of the alarm strongly depends on the organization running the surveillance system, and for this reason this part is not investigated in the paper. Rather we focus on the detection of the alarm event. In case a *face* is *recognized*, a notification FACEID is distributed between the neighbor SSUs.

If there is no correspondence in *selection*, the camera establishes a maximum recognition waiting time (setRecognitionTimeout()): $t_\delta = t_0 + T_{max} - T_{ms} - t_{det}$. Where t_0 is the time moment when the face has been detected in the area, and adds the face features into the reclassification buffer *reclBuffer*. In order to optimize the resources utilization in the network, before to send the recognition request to MS, a camera first sends the recognition request ALARMREQUEST to the neighbors SSUs.

When the reclassification waiting time has passed isTimeOut() and the object is still in the *reclBuffer*, the camera sends classification request ALARMREQUEST to the Main Server entity *MS* and removes the features from the reclassification buffer (*reclBuffer*.remove(*face*)).

At the same time all the SSUs are available to receive messages from the network. In case a message is received, the recipient processes it following the Algorithm 2. In our protocols we consider three types of messages: ALARMREQUEST , ALARMREPLY and FACEID.

The ALARMREQUEST message type corresponds to the messages containing request for a *face* classification. The recipient checks the status of the received *face* in the local subset of a *recognizers* list (getStatus(*face*, *selection*)) and replies ALARMREPLY the results of the recognition *result* to the *sender*.

The ALARMREPLY message corresponds to the reply to a recognition. If recognition results come from MS entity, the recipient integrates (integrate (*recognizers*, *face*)) the recognized face and its status to the local *recognizers* and sends the face notification message FACEIDto its neighbors $view$. Moreover, in case the received status is *alarm* the camera initiates some predefined alarm activity (startAlarmActivity()). In case the recognition results are received from a neighbor camera as a reply for the reclassification, at first the camera increases the counter of replies received for the considered *face* in the local reclassification buffer *reclBuffer*. If the *reclBuffer* contains *face* and the received *status* for the *face* is *null* it means the face has not been recognized by *sender*. In this case, if the *counter* of replies is more or equal to the number of the camera neighbors SSUs, the camera sends the recognition request ALARMREQUEST to the MS and removes the *face* from the reclassification buffer *reclBuffer* (*reclBuffer*.remove(*face*)). In this case the received message means that no one of the neighbor cameras could recognize the face and the help of the MS is needed. If received status is not *null* it means the *face* has been recognized by a sender and the camera integrates it to the local *recognizers*, removes the face from the reclassification buffer and sends the face notification message FACEIDto its neighbors SSUs. In case the received status is *alarm*, the camera also initiates some predefined alarm activity.

Algorithm 2 Passive thread algorithm executed by SSU

```
on event msg < type, face, status, sender > receive do
    if type == ALARMREQUEST then
        result ← getStatus(face, selection);
        send(ALARMREPLY, face, result, sender);

    if type == ALARMREPLY then
        if sender == MS then
            if status == alarm then
                startAlarmActivity();
            integrate(recognizers, face);
            send(FACEID, face, status, view);
        else
            reclBuffer.getFace(face).counter + +;
            if reclBuffer.contains(face) and
            status == null then
                if
                reclBuffer.getFace(face).counter ≥
                view.size then
                    send(ALARMREQUEST, face, null, MS);

                    reclBuffer.remove(face);
            else
                if status == alarm then
                    startAlarmActivity();
                integrate(face, recognizers);
                reclBuffer.remove(face);
                send(FACEID, face, status, view);

    if type == FACEID then
        integrate(face, recognizers);

procedure integrate(recognizers, face)
    if recognizers.contains(face) then
        recognizers.getFace(face).age == 0;
    else
        if recognizers.size ≥ recognizersLimit then
            recognizers.add(face);
        else
            recognizers.remove(recognizers.getOldest());

            recognizers.add(face);
```

FACEID is a message type used for the face features broadcasting to the neighbor SSUs. Hence, whenever a camera receives this type of message it integrates the received face features to the local *recognizers* library.

The integrate(*recognizers*, *face*) procedure is aimed to integrate received/observed face features classes *face* into the local collection *recognizers*. In case the *recognizers* already contains the *face* features, the associated age is set to 0. Otherwise, In case the *size* of the collection has reached the limit *recognizersLimit* the camera swaps the received *face* with the one with the largest *age*. In case there are vacant places in the collection, the camera simply adds the received features in the local collection (*recognizers*.add(*face*)).

5 Performance Evaluation of Classification Algorithms

The problem of recognition using devices with limited computational capabilities is an hot research topics [6, 14, 22]. In this section, we evaluate the performance and accuracy of *1NN* and *weighted kNN* classification Algorithms 2. The results of the tests are used in the following simulation and evaluation of the distributed video surveillance algorithm (Section 6).

In order to compare against most recent face recognition techniques, we also performed the experiments using the weighted kNN algorithm with face features extracted using a Convolutional Neural Network (CNN). We used the 16-layer VGG-Face Deep CNN [25] provided by the Visual Geometry Group of the University of Oxford. However, due to the limitations in the Raspberri Pi resources, the timing for CNN features extraction was too high for the purpose of the distributed approach proposed in this work.

To execute face recognition tests, we used the extended version of the *Yale Database B* [20][1]. From the dataset we have taken 350 images for each face. Among these, 300 are used as test set and the remaining ones are used for the *training set* (TS) of each person. We evaluated the performance of two facial features, LBPH and the Principal Component Analysis (PCA) algorithm applied on the LBPH. For the LBPH features comparison, we have applied the *Chi-Square distance* [2]. In order to evaluate the dissimilarity measure after PCA algorithm applied on the LBPH, we apply the Euclidean distance. Concerning the performance metrics, we consider the *accuracy* and *labelling rate* for the classification algorithms. Accuracy is defined as the ratio between the number of correctly labelled samples and the total number of labelled samples. Labelling rate is defined the ratio between the number of (correctly or incorrectly) classified samples and the total executed classifications.

[1]http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html

According to our evaluations, on the Raspberry PI device the time required by the LBPH feature extraction is $\approx$ 30ms and the χ^2 distance evaluation between two features is about 9ms.

5.1 LBPH Evaluation

1NN Classifiers The confidence for the response of the 1NN classifier is evaluated (1). However, since the confidence is related to the distance of the nearest object $d(q, o_j)$, we evaluate the performance of 1NN classifiers using this distance as the threshold parameter t. In other words, scanning the training set an object o_i will not be considered if $d(q, o_i) > t$.

Figure 2 shows the results of the experiments with 1NN classifiers for different training set sizes (5 and 50 samples per class) and for different values of the distance threshold t in the range $0 - 40$. For increasing values of the threshold t, the classifier is able to label more faces but leading to a lower accuracy. For a threshold lower than 20, an accuracy close to one is achieved. However, for the same threshold and a higher number of samples in the training set, more faces can be classified with an increasing accuracy. For example, if 5 samples for each class are available in the training set and we use a threshold $t = 20$, the classifier is able to label 30% (corresponds to 0.3 labeling ratio on the figure) of faces with an accuracy of 0.9. If the same threshold is used but the number of training samples per class is augmented to 50, the 70% of the faces can be classified with an accuracy of around 0.95. Finally, for higher threshold values, the classifier labels each face while the accuracy remains constant since the distance of the nearest object from the query does not change anymore. Thus, if an accuracy higher than 0.9 is required, a threshold lower than 20 must be used.

Figure 3 relates the accuracy and the ratio of labeled faces considering the number of samples in the training set for each class. The figure shows that more faces can be labeled with higher accuracy when the amount of samples increases. For example, if we want to be able to label 80% of elements (0.8 on the figure), and we need high accuracy (suppose 0.9), 50 samples per class is a good starting point for the training set.

Weighted kNN Classifiers In order to evaluate the weighted kNN classifiers, we analyze two kind of confidence metrics: c_{wkNN_1} and c_{wkNN_2} (defined in

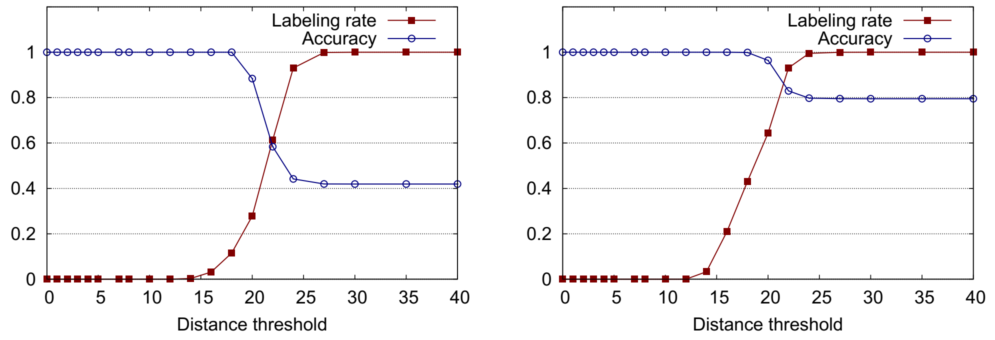


Fig. 2 1NN accuracy and the ratio of labeled faces for different distance thresholds and different training set sizes (from left to right: 5 and 50 samples per class)

Section 2). c_{wkNN_1} is a measure of confidence of the classifier has about the first and the second class with higher score among the others. c_{wkNN_2}, instead, conveys the probability that q belongs to the class with absolute higher score (Fig. 4).

The tuning of k_{opt} depends upon the size of the TS. Table 1 shows the accuracy of the method for different combinations of k and TS sizes. For a small TS, a small value of k gives better performance. If TS grows, the best k grows as well, until it reaches a maximum value of 20. Since the accuracy obtained by these k values is quite the same, we analyze only how the accuracy and the number of labeled elements change if $k_{opt} = 20$, when the confidence c_{wkNN_1} grows from 0 to 1. As we can see on the Fig. 5, the higher is the required confidence, the higher is the accuracy, but the lower is the number of labeled elements. For bigger sizes of TS a higher ratio of elements can be labeled with higher accuracy for a fixed confidence value.

Figure 4 shows the relationship between the obtained accuracy and the ratio of labeled elements for different TS sizes. If we compare Fig. 4 with Fig. 3, weighted kNN classifier is able to label more objects for a given accuracy and a fixed TS size.

Finally, Fig. 6 shows how the accuracy and the ratio of labeled elements change when c_{wkNN_2} is applied. Note that the meaning of $c_{wkNN_2} = 1$ is that all objects of the response set belonging to the winner class c_w are at distance $d = 0$ from the query. Plots are in semi-logarithmic scale since for a low confidence a high accuracy is reached, whereas the number of labeled faces rapidly decreases. Nevertheless, the confidences c_{wkNN_1} and c_{wkNN_2} can be combined in order to have a higher confidence of the response of the classifier.

5.2 PCA on LBPH Feature

Evaluating the LBPH feature on a 100 × 100 pixels image is an easy task also for an embedded node.

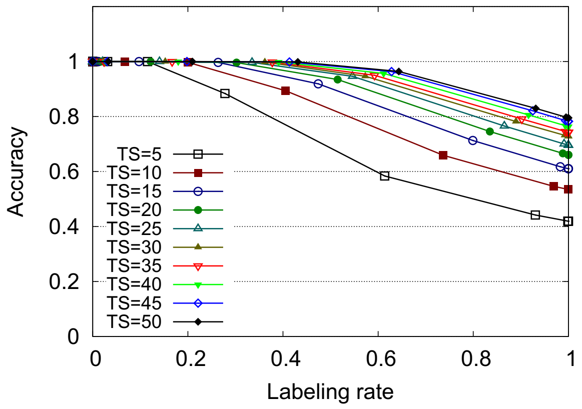


Fig. 3 1NN Accuracy vs Labeling rate for different number of training set samples per class

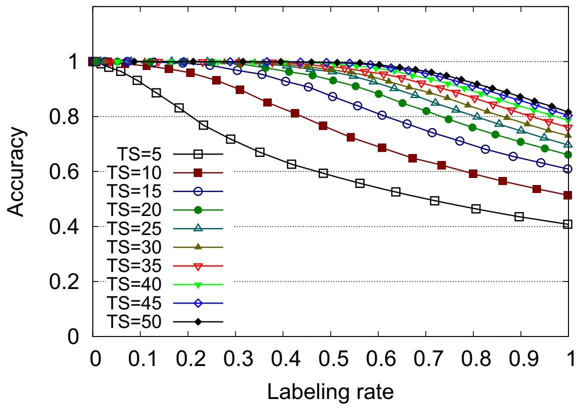


Fig. 4 Weighted kNN with $k_{opt} = 20$ (Accuracy and Labeling rate) for different training set sizes

Table 1 Accuracy achieved by weighted kNN algorithm for different k values and sizes of TS

TS images \ k	*5*	*7*	*10*	*20*	*30*
10	0,5399	**0.54165**	0.53734	0.51384	0,5017
20	0,666	0.66861	**0.66945**	0.66083	0,6509
30	0,7299	0.73425	**0.73522**	0.7298	0,7206
40	0,7709	0.77458	0.78125	**0.78612**	0,7761
50	0,8035	0.80809	0.81129	**0.81602**	0,8108

The best values are in bold

However, the storage used for the TS and the time spent to evaluate the χ^2 distance can still be high for the feature length we considered. Our goal is to be increase the efficiency without affecting the accuracy. Hence, in this section we present the results of the PCA algorithm applied to LBPH feature in order to reduce the feature length [1].

In order to compare the features as dissimilarity function we used the Euclidean distance. In the following, the number of principal components has been fixed to 256.

1NN Figure 7 shows how the accuracy and the labeling rate vary for different TS sizes and distance thresholds. For a threshold higher than 0.3, the accuracy starts to decrease whereas the number of labeled faces increases. For thresholds higher than 1.3, more than the 90% (corresponds to 0.9 on the figure) of faces are labeled and the accuracy is constant. Finally, Fig. 8 shows the relationship between the labeling rate and the accuracy obtained for different TS sizes. A higher number of TS samples allows us to get a higher accuracy.

These results can be compared with the ones we described for standard LBPH feature. The comparison suggests that this method can be applied if accuracy 1 is not required and some misclassification can be accepted.

Weighted kNN As in the case without PCA, the k_{opt} value changes when the TS is enlarged with new samples. Since the variation of the accuracy for k equals 5, 7, and 10 is very low, in this case, we consider how accuracy and the labeling rate change for $k_{opt} = 10$. Figures 9 and 10 show the results of the accuracy and labeling rate for different confidence thresholds and TS sizes. These results can be compared with the weighted kNN algorithm without PCA. Even though high accuracy can be reached, the labeling rate when the accuracy is near 1 is very low. On the other side, in the previous case for weighted kNN on LBPH, when accuracy 1 is required, the labeling rate is near 0.7. Despite the heavy space reduction, the weighted variant of kNN is still able to reach good results and perform much better then other considered classifiers.

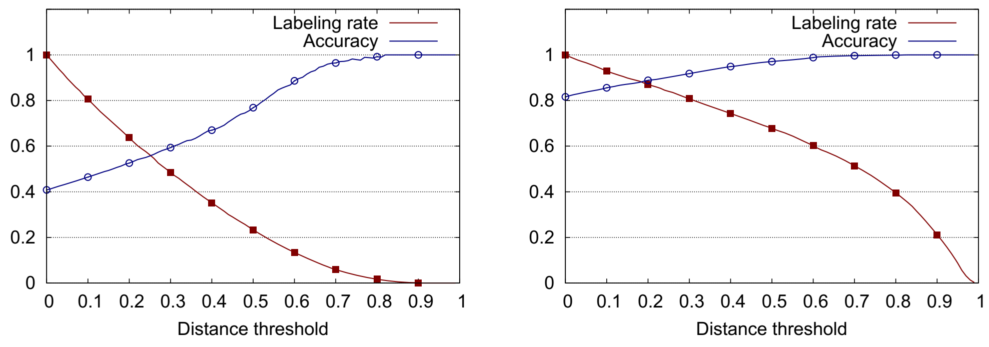


Fig. 5 Weighted kNN accuracy and the ratio of labeled faces for different confidence thresholds(c_{wkNN_1}) and training set sizes(from left to right: 5 and 50 samples per class), $k_{opt} = 20$

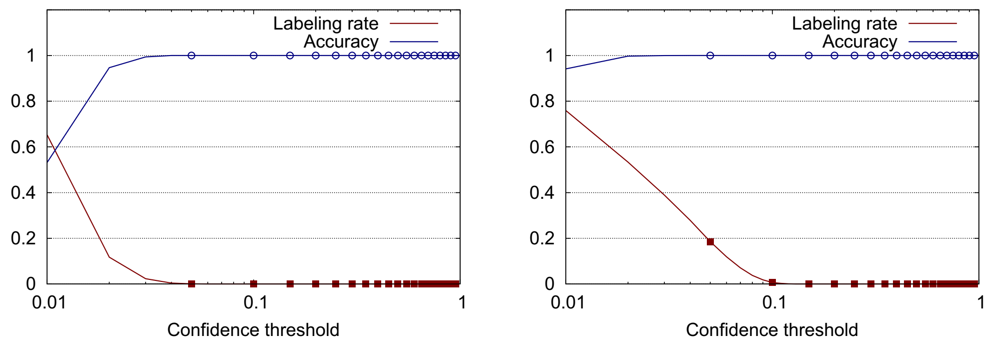


Fig. 6 Weighted kNN accuracy and the ratio of labeled faces for different confidence thresholds (c_{wkNN_2}) and training set sizes (from left to right: 5 and 50 samples per class), $k_{opt} = 20$. Plots are in semi-logarithmic scale

6 Evaluation of the Area Surveillance Algorithm

In this section, we first present the mobility model details and the workload settings we used for the evaluation. Then, we evaluate the performance of the described classifiers in a simulated network of SSUs. The evaluation is based on a simplified network setup and is aimed at analyzing the impact of (i) the SSUs cache size and (ii) the type of classifiers applied at the SSU side. Finally, we evaluate the impact of the proposed distributed surveillance algorithm on the network and *MS* load. Our simulations rely on the PEERSIM simulator [24], a light weight simulator widely used in the community of distributed protocols. To model the SSU hardware, we have considered each SSU to be equivalent to a Raspberry Pi device [33].

6.1 Workload Setup and Surveillance Layout

In order to execute the network simulations, we have used two different system setups. The first, *setup 1* in the following, represents the area covered by a grid of 8 × 8 SSUs where 28 simulated persons are randomly placed. The system knows only 24 faces so that 4 of them are unknown. SSUs' surveillance areas do not overlap each other but the whole surveillance area is covered by them. Initially, for each person, random starting and end points are chosen. During the simulation, persons move towards their final destinations by one step at time following the Random Way Point mobility model [13]. Once they arrive, a new end point is chosen. This operation is performed ten times for each person. In order to simulate the capturing of a person's face by an SSU, we simulate the case in which a person remains in the same position with a relative high probability, so to allow the SSU to detect the face in several sequential frames.

In the second setup, *setup 2* in the following, 625 SSUs are uniformly distributed in the area of 250×250 meters, with an average overlapping of area surveillance with neighbor SSUs of $p_{overlap} = 20\%$. Hence, each SSUs is responsible for an area of about $10m \times 10m$, where 20% is shared with neighbor SSUs. The disjoint of the surveillance areas we simulate via "blind zones" using the probability of the object to be detected in the assigned to the SSU surveillance area. At the start of the simulation each SSU has associated a detection probability that is randomly chosen in the range $p_{detection} = [0.5 - 1]$. For example the SSU with 0.8 probability for object detection simulates the situation where the SSU is able to control 80% of the assign sector and 20% is a blind zone.

Persons in the "surveillance area" move according to a more complex mobility model, adapted in a previous paper [15] for the movements of avatars in a virtual environment. In this model, the area has a number of *hotspots* equals to n_{hs} (Fig. 11a). The hotspots are the most attractive places in the area, for example food court or meeting rooms, so, at any given point in time, it is expected to find more persons there than in other places. We consider the exit and entrance of the whole area to be at the same place, specifically we assume it to be the point with coordinates (0, 0) on the map.

Each person that enters the surveillance area follows the same behavior. First, the person chooses a random number of hotspots to visit and put them in a queue. Second, it moves towards them, starting from the first and proceeding in order, removing an hotspot from the queue once it has been visited. After

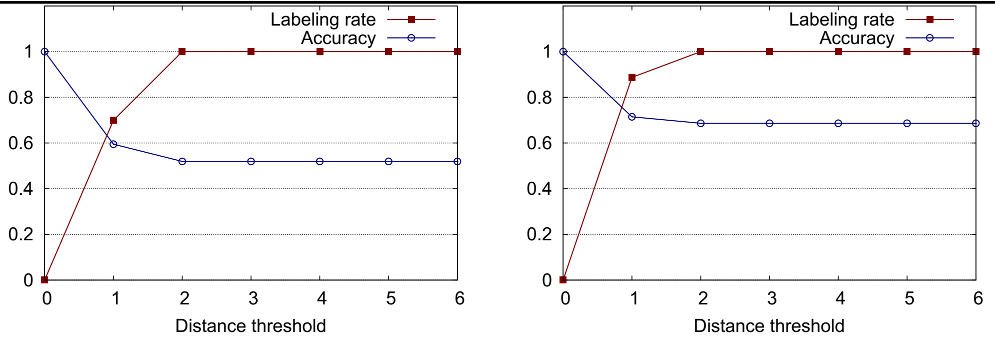


Fig. 7 1NN combined with PCA accuracy and the labeling rate for different confidence thresholds and TS sizes (from left to right: 15 and 50 samples per class)

the queue of the hotspots is empty the person leaves the surveillance area moving towards the exit point. The speed of the person movement in the area v_a is randomly chosen in a predefined range (Table 2). Whenever a person reaches an hotspot it (1) randomly chooses the residence time t_{pause} at the hotspot and (2) changes its speed to a new randomly chosen value. As a reference, all the parameters used in the experiments are listed in Table 2.

For the setup 2, we considered two possible scenarios (Fig. 11). In the first scenario (workload A in Fig. 11b), the population of the area follows a bell-shaped curve very similar to normal distribution in which the population start from 0 and reach a maximum of 800 persons. The second scenario (workload B in Fig. 11c) exhibits a linear growth of the population until the maximum and then keeps the maximum until the end of the simulation. In our simulations, we considered the 10% of persons in the area to be *unknown* (i.e. associated with alarm tag).

6.2 Network Evaluation with Different Classification Algorithms at SSUs for Setup 1

In Section 5, we have shown that the best results in terms of accuracy and the percentage of labeled faces is achieved by the weighted kNN classifier. Here, we apply the weighted kNN classifier algorithm for the recognition of incoming persons at MS and evaluate the effectiveness of SSUs side recognition based on the following classification algorithms: (i) 1NN, (ii) weighted kNN and (iii) a combination of 1NN and weighted kNN. The idea behind the combined algorithm is that 1NN is more robust than weighted kNN for small cache sizes, whereas the latter performs better for greater cache sizes. In the combined algorithm, when a new face must be labeled, the weighted kNN is first applied, then we check if 1NN returns the same label proposed by the weighted kNN. When weighted kNN is used, a threshold on both confidences c_{wkNN_1},

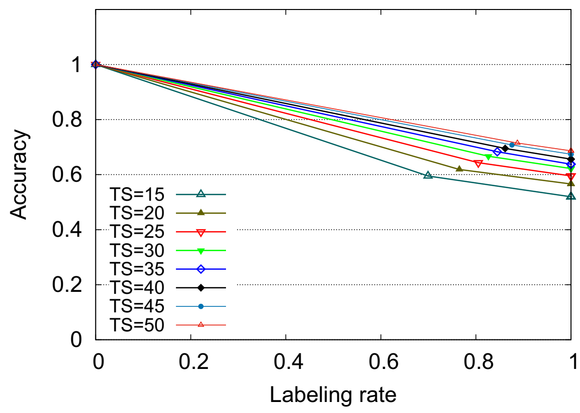


Fig. 8 1NN combined with PCA (Accuracy vs Labeling rate); different TS

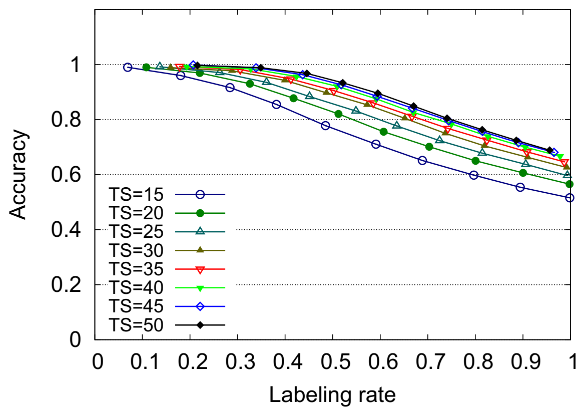


Fig. 9 Weighted kNN combined with PCA (Accuracy and Labeling rate) for different training set sizes and $k = 10$

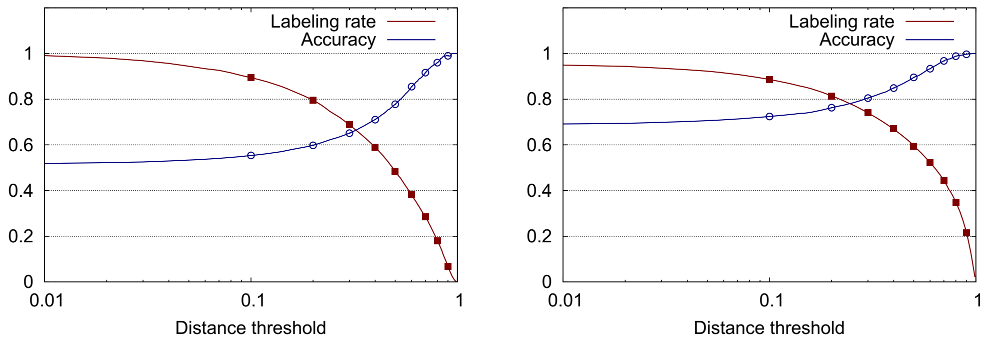


Fig. 10 Weighted kNN combined with PCA accuracy and labeling rate for different confidence thresholds and training set sizes (from left to right: 15 and 50 samples per class), $k_{opt} = 10$. Plots are in semi-logarithmic scale

c_{kwNN_2} (2), (3) is applied, otherwise the algorithm is not able to recognize classify faces if cache size is too small. Finally, note that the confidence in classification on the MS is higher than the one obtain by SSUs, this is because the MS has a virtually infinite cache size and therefore more negative samples.

Figure 12 shows the average number of sent FACEIDand ALARMREQUEST messages by a SSU for different classifiers techniques. The average number of ALARMREQUEST becomes smaller when the number of elements in the caches grows regardless of the classifier used. For small cache sizes, in case of weighted kNN alone, this number is smaller in comparison with other techniques. This is connected with the error committed by this type of classifier when the cache size is small. That is, the new incoming face is often wrong labeled with one of the classes available in cache and no ALARMREQUEST is made. The number of FACEIDmessages sent is comparable for all the classifiers and mostly depends on the recognition performance of the applied classifier technique (Section 5).

Figure 13 shows the evaluation results for the number of received, used, and never used person classes in the cache. As we can see, for bigger cache sizes the percentage of used and not used classes are comparable. This aspect is important, since the nodes memory is not wasted. The results suggests a cache size of (at least) 5 classes to be used.

6.3 Network Performance Evaluation

In order to evaluate the load imposed on the *MS* and the performance of the distributed surveillance system, we executed an extensive simulation of 6 hours using the PEERSIM simulator and the workload mobility models described beforehand (setup 2). We evaluate two algorithms: (1) when all the data is transmitted to the *MS* for the analysis (our baseline) and (2) the adaptive algorithm described in Section 4.

In the baseline model, all SSUs stream the video directly to the *MS*, where the *MS* extracts the face features and recognizes the persons. The baseline curve

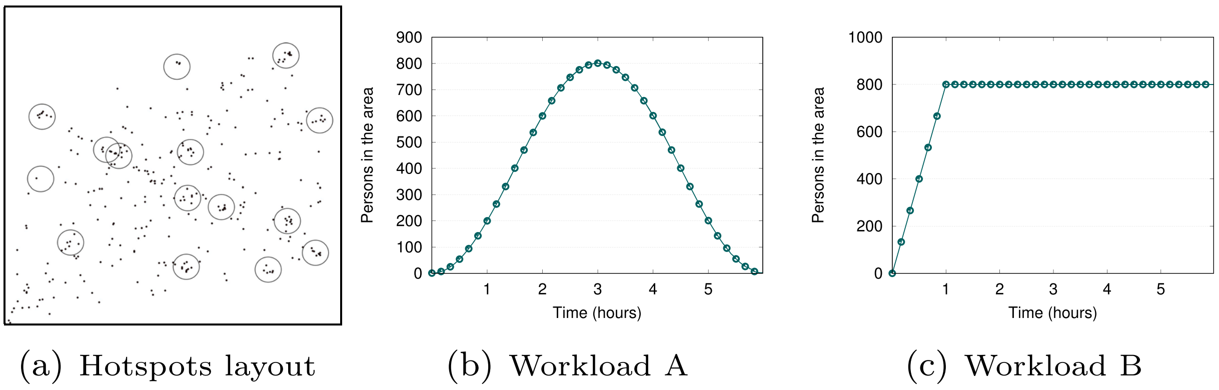


Fig. 11 The area hotspots layout and the workload mobility models with maximum 800 persons in the surveillance area

Table 2 Parameters used in the evaluation

Parameter	Value	Description
N	0-800	Total number of persons in the area
T_{exp}	6 hours	Total simulated time
n_{hs}	15	Number of hotspots in the area
v_a	0-2 m/s	The speed of a person movement between hotspots
t_{pause}	0-50 s	Person residence time at the hotspots
T_{max}	4, 6, 10, 15 s	Maximum allowed recognition delay
ct	45, 135, 360, 450 ms	Time needed to compare the extracted features with one class in the recognition library
n_{SSU}	625 (25 x 25)	Number of SSUs devices in the area
map	250m x 250m	Surveillance area size
$unknown$	10%	Percentage of unknown (alarm tag) persons in the area
$p_{overlap}$	20 %	Overlapping of surveillance spaces of neighbour SSUs
$p_{detection}$	0.5-1	Probability of SSUs to detect a person
$p_{recognition}$	0.5-1	Probability of SSUs to recognize a face
F_{size}	64 K	Size of a vector recognition feature (256*8*8 float)
ΔT	5 s	SSU surveillance area check period

on the Fig. 14 shows the number of recognition the *MS* is required to process in 60sec time interval. Other curves on the figure show the evaluation of the recognition requests to *MS* when the enhanced algorithm is applied and under different system parameters, such as T_{max} and ct. Before to send the recognition request to *MS*, each SSU tries to label the face locally and with the help of the neighbour SSUs. The results demonstrate that the enhanced algorithm can reduce up to 50% of the recognition activity of the *MS* unit in peak hours.

The input T_{max} parameter allows the system administrator to indicate the maximum allowed time between the detection of an object and its actual recognition. The actual T_{max} parameter depends on the scope of the system and can vary between a couple of seconds in case of high security areas up to minutes and even hours. Time interval ct indicates the time needed to compare the extracted face features with one class of the cache recognition library. Each class of the cache is a simulated sample of the face features that characterize a person. Therefore, the time needed by an SSUs to perform a single face recognition task is given by the size of the cache selection times ct. A higher number of the face features in the class increases the accuracy of the recognition, but at the same time increases also the ct.

As we showed in Section 5, a sample of 50 features is enough to achieve a good accuracy for a face recognition. Hence, since as SSU, we considered the Raspberri Pi and based on the comparison time reported in Section 5 on Fig. 14a, we use $ct = 450ms$. As we can see on Fig. 14a, in case of a lower T_{max} the system does not have enough time for the whole

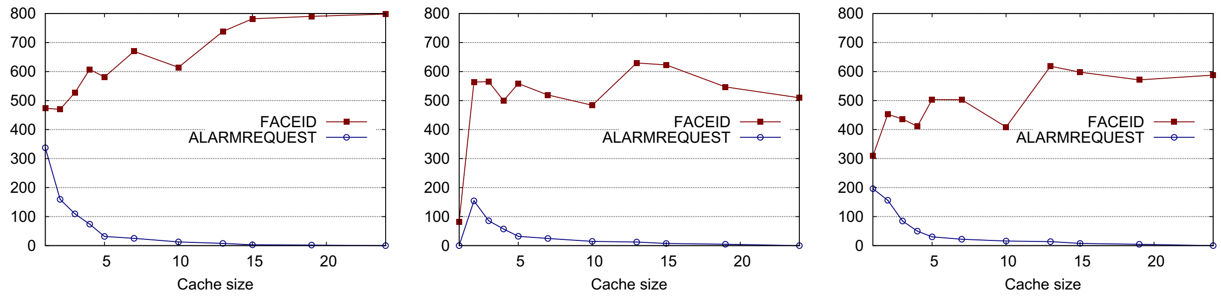


Fig. 12 Average number of FACEIDand ALARMREQUEST messages sent (1NN left, wkNN center and 1NN + wkNN right)

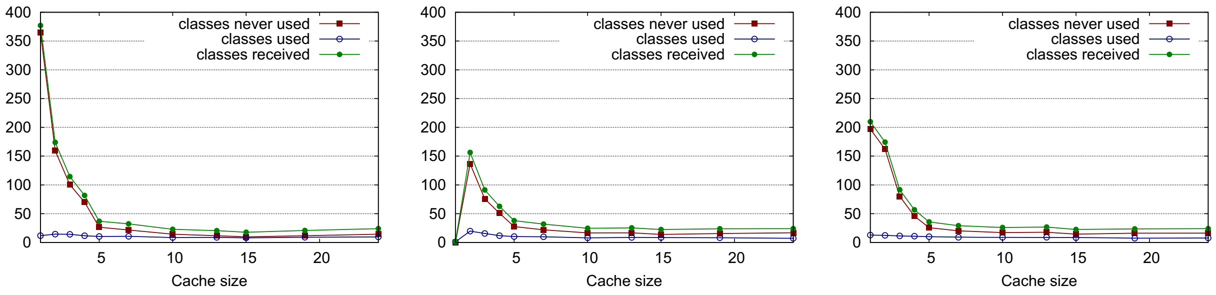


Fig. 13 Average number of used, not used and received classes by nodes (1NN left, wkNN center and 1NN + wkNN right)

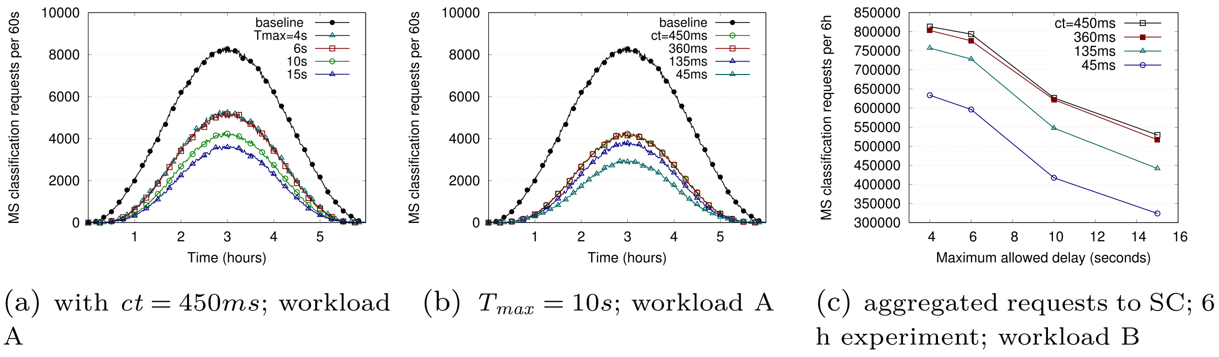
algorithm recognition cycle and it is forced to rely on the *MS* recognition more often. Instead a higher T_{max} allows a SSU to perform the recognition using the cache and to receive the replies from the neighbor SSUs. Nevertheless, even in case of low T_{max}, the enhanced algorithm significantly reduces the recognition load of *MS* compared to the baseline solution.

Figure 14b shows the influence of *ct* time interval on the *MS* recognition load. Even in case of high accuracy of local recognition and relatively low T_{max}, the load of *MS* by recognition requests is significantly lower than the baseline solution. Moreover, lower *ct* values further decreases the load imposed on the *MS* by the recognition. Figure 15 shows the impact of the system parameters on the algorithm effectiveness. Lower values of *ct* and larger of T_{max} significantly reduce the *MS* recognition load. One of the straightforward way to minimize *ct* time is to reduce the accuracy of local recognition algorithm by decreasing the number of features in the sample. In case fast person recognition is not a requirement, the increment of T_{max} can also reduce the load of the *MS*.

We also evaluate the actual face recognition delays in the area. Figure 15 presents the snapshots of the average delays per SSUs at 1, 3 and 5 hours of the experiment with workload A. We can observe that the algorithm timeout restriction is able to guarantee the 100% of recognition in time ($T_{max} = 10s$ and $CT = 450ms$ are used).

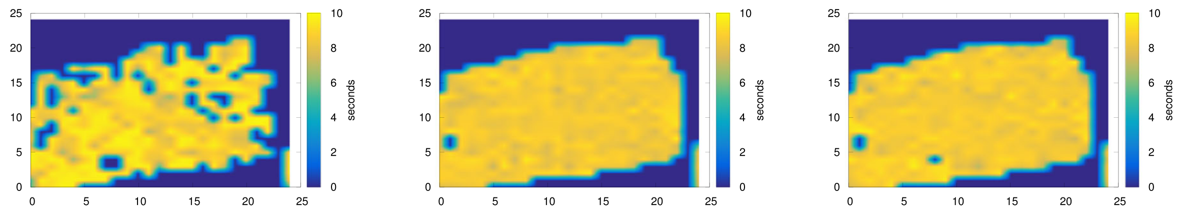
One of the issues of the network communications is the risk to overload the network due to many message exchanges. We evaluate the bandwidth consumption under different network workloads and system parameters (Fig. 16). In our evaluation we consider each of the face to be described with a vector F_{size} =64KB ($256 * 8 * 8$ floats).

Figure 16 present the maximum and the average bandwidth consumption in MB/sec during the time for the workload A (Fig. 16a) and workload B (Fig. 16b) with $ct = 450$ms and $T_{max} = 10$sec. For both workloads the average bandwidth consumption values are relatively small, while the maximum bandwidth consumption at some SSUs reaches the values of more than 2.5MB/sec. As we can see, the difference



(a) with $ct = 450ms$; workload A (b) $T_{max} = 10s$; workload A (c) aggregated requests to SC; 6 h experiment; workload B

Fig. 14 Main Server recognition requests per 60sec for different system parameters



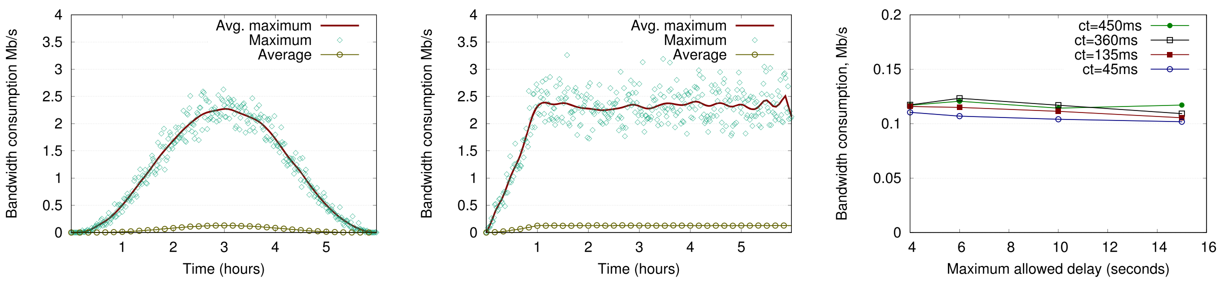
(a) 1st hour of the experiment (b) 3rd hour of the experiment (c) 5th hour of the experiment

Fig. 15 Recognition delay heatmap for a workload A, $ct = 450ms$, $T_{max} = 10s$

between maximum and average values in the area is large. This indicates a high heterogeneity in the system load, that is connected with the presence of hotspots in the area. The hotspots are more visited than other places and this causes a high rate of recognition tasks on the surveillance SSUs. This can also be observed by the snapshots of the area upload bandwidth consumption that are presented on Fig. 17. The Fig. 17 show the map of the bandwidth consumption in the system at 1, 3 and 5 hours of the experiment with the workload A. As you can see the most intensive bandwidth consumption is on the hotspots (Fig. 11a), whereas the other areas of the map are significantly less loaded.

Figure 16c shows the impact of ct and T_{max} parameters on the average bandwidth consumption for a SSU. As we can see, the higher T_{max} and lower ct do not significantly influence on the bandwidth load. We explain it with the fact that since the considered T_{max} time intervals are relatively small, a SSU does not have enough time for a full scan of the local cache. Hence, considering the logic of the algorithm, it sends the outcoming recognition requests to the neighbor SSUs and eventually to MS. In the peak hours of the simulation, especially for the SSUs controlling the hotspots areas, the rate of the outcoming recognition requests to the neighbor SSUs significantly increases. Most of SSUs have around 8 neighbor SSUs in our simulation and the outcoming recognition traffic to these neighbor SSUs is at least 8 times higher than the one to MS. Moreover, the SSU in the system also have to reply to the incoming recognition requests. Hence, the strongest impact on the SSUs bandwidth consumption is due to the neighbor SSUs recognition loop in the algorithm. On the one side, FACEID broadcasting allows us to increase the probability the face to be recognized locally by a SSU in the next surveillance zones. On the other side, as it is shown in the Figs. 17 and 16c, it creates an excessive SSUs bandwidth consumption. However, the FACEID broadcasting can be reduced by applying existing algorithms for the prediction of person trajectory [36].

The results of our simulation show that in order to optimize the resources utilization the network of the surveillance devices should be composed with heterogeneous devices. In other words, the hotspots area and the areas that connect them should be served by more powerful SSUs or even servers, while the surveillance of other areas (like dark sectors on the Figs. 15 and 17) can be maintained by more economic solutions. Another interesting solution could be to apply



(a) workload A; (b) workload B; (c) workload B

Fig. 16 Bandwidth consumption in Mb/s for $T_{max} = 10s$ and $ct = 450ms$ system parameters

a combined approach of both enhanced and baseline algorithms. In other words, whenever a SSU reaches its upload bandwidth consumption limit, for example in case of too many persons in the surveillance area, it can start to stream the video directly to the *MS*.

7 Related Work

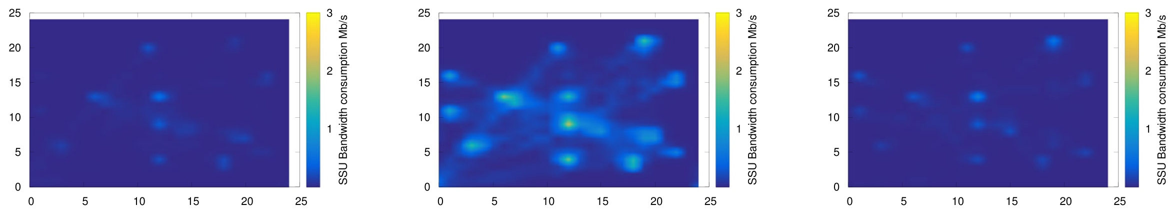
There is a vaste literature on video surveillance system based on distributed cameras, we restrict ourselves to those works most relevant for ours. For a more extensive review on this subject, we refer to the recent surveys [30, 35].

The problem of monitoring a large surveillance space through a distributed face detection and recognition system has been treated in [23]. The work proposes a novel video Surveillance application using HDFS and Map Reduce framework that collects video from multiple cameras from surveillance scenario into cloud for detection and tracking of persons over space and time. The proposed architecture, however, does not rely on smart cameras for the task of face recognition, but rather on a distributed processing infrastructure using map-reduce framework over Hadoop.

The idea of exploiting the computation capabilities and the topology of a distributed network of smart cameras in order to reduce the amount of messaging has been proposed in [38] and [42]. In both approaches the communication is efficiently handled using a task-oriented node clustering that partition the network in different groups according to the pathway among cameras. The former work, however, is limited to face tracking, while second one targets collaborative person tracking with a combination of hardware acceleration and middleware. These works focus on people tracking rather than recognition and use an efficient camera clustering protocol to dynamically form groups of cameras for in-network tracking of individual persons.

Self-coordination of objects handover in a distributed network of embedded smart camera using a knowledge previously detected in neighbor cameras has been done in [19, 28, 39]. In [34], a P2P multi-camera system for multiple object tracking is presented. The data from multiple features are exchanged between adjacent cameras (with partially overlapping fields of view) for object matching. This work does not provide much implementation details, such as the embedded platform used, programming languages, etc. The paper [26] presents an agent-oriented middleware for distributed smart cameras, which supports collaborative image processing. The work is focused on dynamic loading/migration of processing tasks and collaborative work between nodes at the physical layer. Similarly, in [9], based on self-interested autonomous agents, the cameras learn a vision graph describing the spatial relationships between their fields of view (FOVs). Based on such a vision graph, cameras reduce their communication without significantly sacrificing tracking performance. Having the object detected within its FOV, the camera can try to get the responsibility to track the object by bidding for it. This work has been extended in [21], where six different behavioral strategies were available to cameras. A similar problem to the one we study is the *re-identification* problem, which aims at locating moving targets or at determining co-occurrences of events in the FOVs. With this approach, (e.g., see [8] and [29]) the topology of a set of disjoint cameras can be found by analyzing motion patterns at event co-occurrences. Re-identification problem in large distributed camera networks is also addressed in [17]. This work also deal with the general problem of limited resource management, in which, like us, the resources are limited by the maximum number of classifiers and the storage capacity required to store them. Gheissari et al. [12] present a new approach for the person reidentification problem over a smart camera network, based on the fact that people appearance in public spaces is almost the same in a given temporal window. Thus, instead of the face or other passive biometrics, they consider the use of the overall appearance of the individual. In order to do that, the system is be able to establish correspondence between parts and generate signatures that are invariant to changes of illumination, pose and dynamic appearance of clothing.

Concerning the specific problem of face recognition in distributed camera networks, [18] described the design and implementation of a distributed real-time face recognition system using a network of embedded cameras. However, the system is only partially distributed, since a base station that acts as central server is needed to perform the recognition task, while the cameras only deal with the face detection task. A Dynamic Bayesian Network based multi-camera face recognition algorithm is proposed in [3], which



(a) 1st hour of the experiment (b) 3rd hour of the experiment (c) 5th hour of the experiment

Fig. 17 Bandwidth consumption heatmap workload A, $T_{max} = 10s$ and $ct = 450ms$

tries the exploit temporal information among adjacent frames to establish the person-specific dynamics to improve the recognition performance. Also in this work the face recognition is performed in a centralized fashion, and no spatial information about the camera location where the face are detected in exploited.

A fully distributed face recognition system is presented in [11]. In this work authors present a scenario where each node contains a camera of limited aperture, whose images are matched against a randomly selected feature of each face in a statically defined training set. In another similar work [27], a multi-camera face recognition framework using cameras that perform recognition cooperatively was proposed. The main objective of this work was to introduce a method of exploiting spatio-temporal correlations between recognitions in different cameras. In both works, however, the cameras have a complete knowledge of the database used for face recognition, which poses a real obstacle to scalability. In [40] a distributed sensor network (DSN) capable of performing reliable recognition targeted at multiple humans in the indoor environments is presented. In this architecture, the cameras do not perform face reconsign autonomously but rely on a specific node of the DSN.

8 Conclusion and Future Work

In this paper, we propose a distributed camera-aided protocol for area video surveillance based on image classification. In order to minimize the classification load on the main server, recognition tasks take place on the SSUs when possible. To perform person recognition, a SSU uses the local resources together with the resources of the neighbor SSUs. The surveillance devices fall back to the main server when the classification cannot be done in the desired time interval.

We have evaluated two different classification algorithms (1NN and weighted kNN) implemented with LBP. Among the considered classifiers the best performance results are obtained by the weighted kNN algorithm. Therefore, we have evaluated the load of *MS* and network characteristics based on weighted kNN classifiers applied on SSUs. Our evaluation simulated the surveillance area by considering a number of hotspots. The bandwidth consumption of the SSUs that are responsible for the surveillance of the hotspots resulted much higher than for the other SSUs. This suggests to use more powerful devices for the hotpots surveillance, or to apply a combined approach in which the system switches to a pure client-server model recognition and streams video directly to the main server for processing in case of high people concentration at one surveillance spot. Moreover, a further optimization of the bandwidth consumption could be achieved by applying algorithms for the people trajectory estimations directly inside the SSUs, in order to reduce the features broadcasting between neighbor SSUs.

Finally, the evaluation shows that partially placing the recognition tasks on the local resources of surveillance devices can reduce the load on the main server up to 50%. As a consequence, it is possible to run a surveillance service by employing less powerful centralized servers, or, alternatively, a large server can be re-used for the other tasks in the less busy hours.

Acknowledgements This work has been partially funded by the DIITET Department of CNR, in the framework of the ”Revenue Energy and ICT for Sustainability Energy” project.

References

1. Ahonen, T., Hadid, A., Pietikäinen, M.: Face recognition with local binary patterns. In: Computer Vision-eccv 2004, pp. 469–481. Springer (2004)
2. Ahonen, T., Hadid, A., Pietikäinen, M.: Face description with local binary patterns: Application to face recognition. IEEE Trans. Pattern Anal. Mach. Intell. **28**(12), 2037–2041 (2006)
3. An, L., Kafai, M., Bhanu, B.: Face recognition in multi-camera surveillance videos using dynamic Bayesian network. In: Sixth International Conference on Distributed Smart Cameras, ICDSC (2012)
4. Carlini, E., Lulli, A., Ricci, L.: dragon: Multidimensional range queries on distributed aggregation trees. Futur. Gener. Comput. Syst. **55**, 101–115 (2016)
5. Carlini, E., Ricci, L., Coppola, M.: Flexible load distribution for hybrid distributed virtual environments. Futur. Gener. Comput. Syst. **29**(6), 1561–1572 (2013)
6. Cui, L., Yang, S., Chen, F., Ming, Z., Lu, N., Qin, J.: A survey on application of machine learning for internet of things. Int. J. Mach. Learn. Cybern. **9**(8), 1399–1417 (2018)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley (2012)
8. Erdem, U.M., Sclaroff, S.: Event prediction in a hybrid camera network. ACM Trans. Sensor Netw. (TOSN) **8**(2), 16 (2012)
9. Esterle, L., Lewis, P.R., Yao, X., Rinner, B.: Socio-economic vision graph generation and handover in distributed smart camera networks. ACM Trans. Sensor Netw. (TOSN) **10**(2), 20 (2014)
10. Etemad, K., Chellappa, R.: Discriminant analysis for recognition of human face images (invited paper). In: AVBPA, Volume 1206 of Lecture Notes in Computer Science, pp. 127–142. Springer (1997)
11. Gaynor, P., Coore, D.: Distributed face recognition using collaborative judgement aggregation in a swarm of tiny wireless sensor nodes. In: SoutheastCon 2015, pp. 1–6 (2015)
12. Gheissari, N., Sebastian, T.B., Hartley, R.: Person reidentification using spatiotemporal appearance. In: Proc. of the 2006 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, CVPR '06, pp. 1528–1535. IEEE Computer Society, Washington, DC (2006)
13. Hong, X., Gerla, M., Pei, G., Chiang, C.-C.: A group mobility model for ad hoc wireless networks. In: Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 53–60. ACM (1999)
14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Mobilenets, H.Adam.: Efficient convolutional neural networks for mobile vision applications. ArXiv:1704.04861 (2017)
15. Kavalionak, H., Carlini, E., Ricci, L., Montresor, A., Coppola, M.: Integrating peer-to-peer and cloud computing for massively multiuser online games. Peer-to-Peer Netw Appl (PPNA), 1–19 (2013)
16. Kavalionak, H., Gennaro, C., Amato, G., Meghini, C.: Dice: A distributed protocol for camera-aided video surveillance. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), pp. 477–484. IEEE (2015)
17. Kenk, V.S., Kovačič, S., Kristan, M., Hajdinjak, M., Perš, J., et al.: Visual re-identification across large, distributed camera networks. Image Vis. Comput. **34**, 11–26 (2015)
18. Kulathumani, V., Parupati, S., Ross, A., Jillela, R.: Collaborative face recognition using a network of embedded cameras. In: Distributed Video Sensor Networks, pp. 373–387. Springer (2011)
19. Kushwaha, M., Koutsoukos, X.: Collaborative 3d target tracking in distributed smart camera networks for wide-area surveillance. J. Sensor Actuat. Netw., **2**(2) (2013)
20. Lee, K.-C., Ho, J., Kriegman, D.J.: Acquiring linear subspaces for face recognition under variable lighting. IEEE Trans. Pattern Anal. Mach Intell. **27**(5), 684–698 (2005)
21. Lewis, P.R., Esterle, L., Chandra, A., Rinner, B., Yao, X.: Learning to be different: Heterogeneity and efficiency in distributed smart camera networks. In: IEEE 7th Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO), pp. 209–218. IEEE (2013)
22. LiKamWa, R., Hou, Y., Gao, J., Polansky, M., Zhong, L.: Redeye: Analog convnet image sensor architecture for continuous mobile vision. In: Proceedings of the 43rd International Symposium on Computer Architecture, ISCA '16, pp. 255–266. IEEE Press, Piscataway (2016)
23. Mishra, R., Kumar, P., Chaudhury, S., Indu, S.: Monitoring a large surveillance space through distributed face matching. In: 2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), pp. 1–5. IEEE (2013)
24. Montresor, A., Jelasity, M.: PeerSim: A scalable p2p simulator. In: Proc. of P2P'09, pp. 99–100. IEEE (2009)
25. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: British Machine Vision Conf. (2015)
26. Quaritsch, M., Rinner, B., Strobl, B.: Improved agent-oriented middleware for distributed smart cameras. In: ICDSC'07. First ACM/IEEE Int. Conf. on Distributed Smart Cameras, 2007, pp. 297–304. IEEE (2007)
27. Rambach, J., Huber, M.F., Balthasar, M.R., Zoubir, A.M.: Collaborative multi-camera face recognition and tracking. In: 2015 12th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6 (2015)
28. Saini, M.K., Atrey, P.K., Saddik, A.E.: From smart camera to smarthub: Embracing cloud for video surveillance. Int. Journal of Distrib. Sensor Networks (2014)
29. SanMiguel, J.C., Micheloni, C., Shoop, K., Foresti, G.L., Cavallaro, A.: Self-reconfigurable smart camera networks. Computer **47**(5), 67–73 (2014)
30. Song, M., Tao, D., Maybank, S.J.: Sparse camera network for visual surveillance–a comprehensive survey. ArXiv:1302.0446 (2013)
31. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition, CVPR '14, pp. 1701–1708. IEEE Computer Society, Washington, DC (2014)
32. Turk, M., Pentland, A.: Eigenfaces for recognition. J Cogn. Neurosci. **3**(1), 71–86 (1991)

33. Upton, E., Halfacree, G.: Raspberry Pi User Guide. Wiley (2012)
34. Velipasalar, S., Schlessman, J., Chen, C.-Y., Wolf, W.H., Singh, J.P.: A scalable clustered camera system for multiple object tracking. EURASIP J. Image Video Process. **2008**, 22 (2008)
35. Wang, X.: Intelligent multi-camera video surveillance: A review. Pattern Recogn. Lett. **34**(1), 3–19 (2013)
36. Wiest, J., Hoffken, M., Kresel, U., Dietmayer, K.: Probabilistic trajectory prediction with gaussian mixture models. In: 2012 IEEE Intelligent Vehicles Symposium (IV), pp. 141–146 (2012)
37. Wiskott, L., Fellous, J.-M., Kruger, N., von der Malsburg, C.: Face recognition by elastic bunch graph matching. In: ICIP (1), pp. 129–132 (1997)
38. Yoder, J., Medeiros, H., Park, J., Kak, A.C.: Cluster-based distributed face tracking in camera networks. IEEE Trans Image Process **19**(10), 2551–2563 (2010)
39. Yonga, F., Junior, A.G., Mefenza, M., Saldanha, L., Bobda, C., Velipassalar, S.: Self-coordinated target assignment and camera handoff in distributed network of embedded smart cameras. In: Proc. of the Int. Conf. on Distributed Smart Cameras, p. 16. ACM (2014)
40. Yun, S.S., Nguyen, Q., Choi, J.: Distributed sensor networks for multiple human recognition in indoor environments. In: 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 753–756 (2016)
41. Zafeiriou, S., Zhang, C., Zhang, Z.: A survey on face detection in the wild: Past, present and future. Comput. Vis. Image Underst. **138**, 1–24 (2015)
42. Zarezadeh, A., Bobda, C., Yonga, F., Mefenza, M.: Efficient network clustering for traffic reduction in embedded smart camera networks. J. Real-Time Image Proc., 1–14 (2015)