Video Analysis in PTZ Camera Networks

From master-slave to cooperative smart cameras

Christian Micheloni*, *Member, IEEE*, Bernhard Rinner†, *Senior Member, IEEE*, Gian Luca Foresti*, *Senior Member, IEEE*

* Department of Computer Science, University of Udine, Udine ITALY

† Institute of Networked and Embedded Systems, Klagenfurt University, Klagenfurt AUSTRIA

Abstract—**Pan-Tilt-Zoom (PTZ) cameras are able to dynamically modify their field of view. This functionality introduces new capabilities to camera networks such as increasing the resolution of moving targets and adapting the sensor coverage. On the other hand, PTZ functionality requires solutions to new challenges such as controlling the PTZ parameters, estimating the ego-motion of the cameras and calibrating the moving cameras.**

This tutorial provides an overview of the main video processing techniques and the currents trends in this active field of research. Autonomous PTZ cameras mainly aim to detect and track targets with the largest possible resolution. Autonomous PTZ operation is activated once the network detects and identifies an object as sensible target and requires accurate control of the PTZ parameters and coordination among the cameras in the network. We, therefore, present cooperative localisation and tracking methods, i.e., multi-agent and consensus-based approaches to jointly compute the target's properties such as ground-plane position and velocity. Stereo vision exploiting wide baselines can be used to derive 3D target localisation. This tutorial further presents different techniques for controlling PTZ camera hand-off, configuring the network to dynamically track targets and optimizing the network configuration to increase coverage probability. It also discusses implementation aspects for these video processing techniques on embedded smart cameras—with a special focus on data access properties.

I. INTRODUCTION

The progress in sensors and computing leveraged the development of novel camera networks and processing frameworks. In particular, small to medium video surveillance networks have been deployed in different environments such as airports, train stations, public parks and office buildings [17].

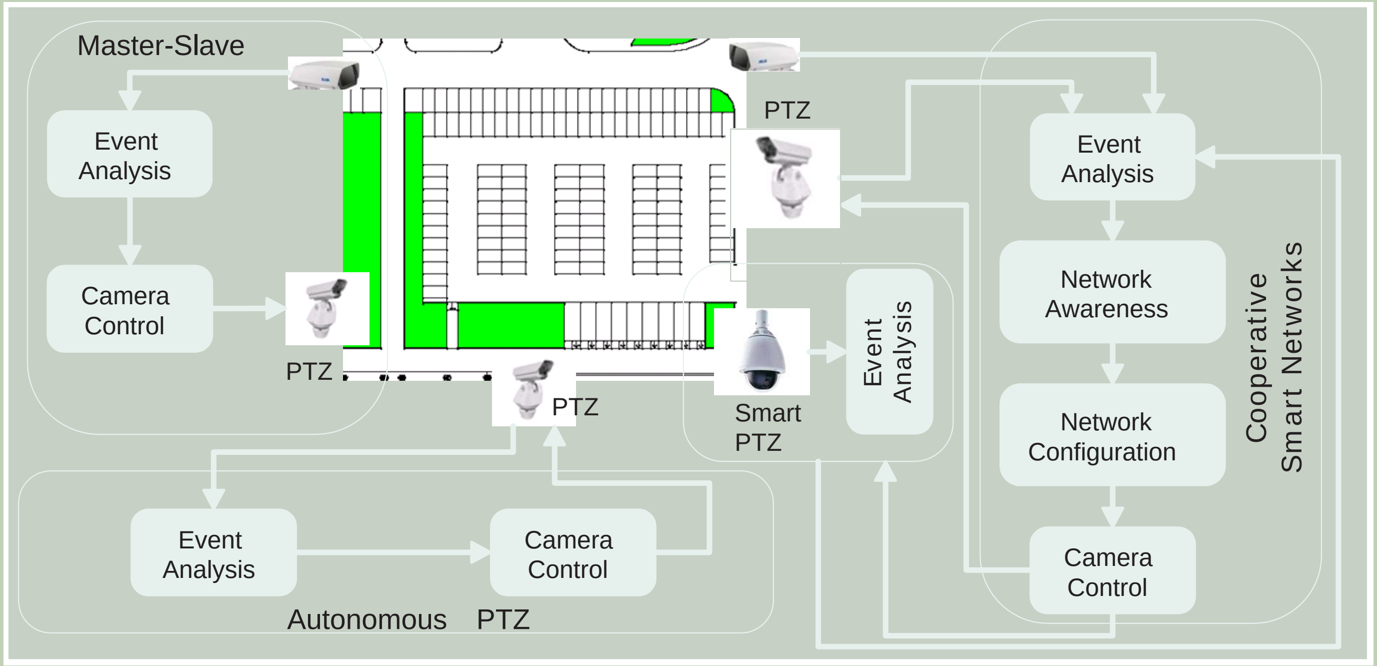
The availability of new information guided researchers in developing new intelligent surveillance systems [38], in which dozens of cameras cooperate to solve complex tasks such as the tracking of moving objects. Originally, such a task had been tackled by single cameras. The goal was to achieve a consistent detection of the track and preservation of correct labelling in single and multiple object scenarios, respectively.

In modern surveillance systems, one of the major challenges in multi-camera tracking is the consistency of the objects' labelling through different fields of view (FOVs). Multi-sensor calibration techniques for the transition between the cameras' views or the development of multi-target-multi-sensor tracking mechanisms have been the research core for such kind of networks [5], [10], [23].

Static camera networks, even though they allow to cover a wide area for monitoring activities, have different limitations. Normally, camera deployment has to consider overlapping the individual FOVs. Image resolutions are usually low due to the wide angle view adopted to increase the monitored area. When no overlapping FOVs are present, signal processing has still to be performed on a single data source. As a consequence, occlusions cannot be solved using multi-sensor based techniques. Localisation in the real world is limited by ground-plane constraints affecting the performance when lower sections of objects are occluded. All these problems can be reduced if any point in the environment belongs to two or more FOVs of the cameras. Naturally, such a constraint poses severe limitations for networks covering large areas due to the huge amount of video sensors needed.

The introduction of *Pan-Tilt-Zoom (PTZ) cameras* brought new capabilities to surveillance networks as well as new problems to be solved. In particular, PTZ cameras can adapt their FOVs which can be exploited to focus the video-network attention on areas of interest. Hence, some problems related to non-overlapping FOVs can be overcome (e.g., occlusions, localisation and low target resolution). On the other hand, the PTZ movement has introduced the non trivial ego-motion estimation problem. Thus, well known low-level techniques such as motion detection and calibration needed further development to be feasible for these new moving sensors.

This tutorial provides a comprehensive introduction of PTZ camera networks for active surveillance. We focus on the different signal processing methods to achieve robust object tracking and localisation by means of PTZ cameras' cooperation and reconfiguration. The tutorial is organized to show the evolution of the signal processing techniques from single autonomous PTZ cameras to more complex cooperative smart networks (see Fig. 1). It covers low-level techniques for autonomous object detection as well as more high-level methods for the cooperation of different moving cameras. In particular, the analysis of the localisation performance of static and PTZ cameras networks is discussed. The dynamic configuration of the PTZ cameras is further described to propose a novel research stream focusing on the computation of the optimal configuration of the PTZs network to improve coverage, hand-off and tracking. We then present techniques for embedding such algorithms into smart cameras to realize new autonomous and distributed video networks which finally results in a discussion on the future of smart reconfigurable sensor networks.



The simplest usage of PTZ cameras is the master-slave interaction. The network composed by static cameras performs all the processing for the event analysis and computes the camera motion parameters to direct the PTZ gaze onto the object of interest. To introduce autonomous PTZ cameras different active vision techniques can be applied to make the moving camera more intelligent. It is then possible to locally process the video stream to extract useful information about the activities occurring within the monitored scene. By exploiting active vision techniques together with common static camera networks it is possible to realize cooperative frameworks in which the analysis is performed by processing both streams coming from PTZ and static cameras. More, the ability of PTZ cameras to modify their FOV on the basis of the system needs (e.g. higher resolution and more FOV overlap) allows to define ad hoc stereo vision systems, dynamic hand-over areas and so on. The development of such techniques on-board of modern smart cameras enables to consider different parameters (e.g. area coverage, power consumption and bandwidth usage) to compute the optimal configuration of the network in terms of number of sensors switched on and their configuration (PTZ values).

Fig. 1. Different architectures for networks exploiting PTZ cameras: master-slave, autonomous PTZ and cooperative smart cameras

II. Autonomous PTZ cameras

Aloimonos has been the first to propose the *active vision* paradigm to describe the dynamic interaction between the observer and the object observed to actively decide what to see [2]. In this way, the video sensor can be controlled to keep the gaze on a selected target. Such a simple idea brought a new set of problems that in context of static cameras were inexistent. A first problem is represented by the fact, that in context of moving cameras, even pixels belonging to static objects appear to move in the camera frame. Such an effect is called ego-motion and its estimation and compensation represented the first objective of the active vision research area.

A. Ego-Motion Estimation

Following the active vision paradigm, Murray and Basu [34] investigated the problem of tracking moving objects by means of a Pan-Tilt camera. A first solution to the estimation of the ego-motion had been given. An analytic determination of the image transformation for the registration of the images was proposed. Such a transformation allows to compensate the ego-motion when a camera is rotating along the optical centre. In other words, the registration problem consists in determining a transformation $\mathbf{T}$ such that for two time related frames $I(\mathbf{x}, t)$ and $I(\mathbf{x} + \delta\mathbf{x}, t + \tau)$ the following equation holds:

$$I(\mathbf{x}, t) = \mathbf{T} \cdot I(\mathbf{x} + \delta\mathbf{x}, t + \tau).$$

When different camera motions are considered, different registration techniques, thus different transformations $\mathbf{T}$, have to be taken into account to model the ego-motion effects. A survey on image registration techniques [50] explains how particular transforms (translational, affine, perspective, etc.) have to be computed for registering images. Irani and Anandan [19] addressed the problem of detecting moving objects by using a direct method which estimates a ”dominant” 8-parameters of an affine transformation. Araki *et al.* [3] proposed to estimate the background motion by using a set of features and an iterative process ($LSMedS$) to solve the overdetermined linear system. All these systems, feature based or linear, have to cope with the outliers detection to filter out bad tracked features or mismatched points [16]. Micheloni and Foresti [32] developed a new feature rejection rule for deleting outliers, based on a feature clustering technique [15]. Once the images are registered and ego-motion compensated (i.e., static pixels overlap), the moving objects can be extracted with frame-by-frame differencing techniques. Though, when zoom operations are needed to focus on an object of interest, such techniques do not guarantee robust results.

B. Fixation

Rather than computing and compensating the ego-motion during zoom operations, a better solution is to directly track the target by adopting a fixation point to be kept during zoom. Tordoff and Murray [46] solve the problem of tracking a fixation point, mainly the centre of mass of an object, by adopting an affine projection. In particular, when the distance between the object and the camera is much greater than the depth of the object (as it often happen in video surveillance applications) it is possible to consider the following affine projection:

$$\mathbf{x} = \mathbf{M}\mathbf{X} + \mathbf{t} \tag{1}$$

where $\mathbf{x}$ is the fixation point, M is the affine transform, X is the real position of the point and $\mathbf{t}$ is a translation vector.

1) Computing the fixation point: Considering four coplanar points A, B, C, O belonging to the same object and defining a vector space $\bar{B} = \{A-O, B-O, C-O\}$, we can determine all other points belonging to the object or to the space. Moreover, a generic point X can be defined by means of the three affine coordinates α, β, γ as follows:

$$X = \alpha(A-O) + \beta(B-O) + \gamma(C-O) + O \tag{2}$$

Given two different views (e.g. two consecutive frames of a moving camera) for the four points (a, b, c, o and a', b', c', o') it is possible to compute the affine coordinates of a fifth point X by solving the overdetermined linear system:

$$\begin{bmatrix} x-o \\ x'-o' \end{bmatrix} = \begin{bmatrix} a-o & b-o & c'-o' \\ a'-o' & b'-o' & c'-o' \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \tag{3}$$

It is worth noticing that after having computed the coordinates of the point X we do not need to compute it again in a new view since it is given by:

$$x'' = \alpha(a''-o'') + \beta(b''-o'') + \gamma(c''-o'') + o'' \tag{4}$$

Therefore, during the tracking phase, supposing that the fixation point has a linear transformation from position $\mathbf{g}$ on time instant t to the position $\mathbf{g}'$ on time instant $t+1$, we can compute its affine coordinates $[\alpha_g, \beta_g, \gamma_g]^T$ by adopting equation (3). On a further frame $t+2$, the four points are projected in new positions respectively a'', b'', c'', o'' and the equation (4) will give the new position of the fixation point $\mathbf{g}''$. The main issue in adopting this technique concerns the determination of the points to use during the computation of the affine transform.

2) Clustering feature points: To solve this problem it is possible to adopt a feature clustering technique able to classify a tracked feature as belonging to different moving objects or to the background. The objective is first to identify and compensate the motion induced by the zoom, if present, then to apply a clustering technique to identify the features belonging to objects with different velocities. In this way, all the features belonging to static objects after zoom compensation have a null displacement. The resulting algorithm is showed in Algorithm 1.

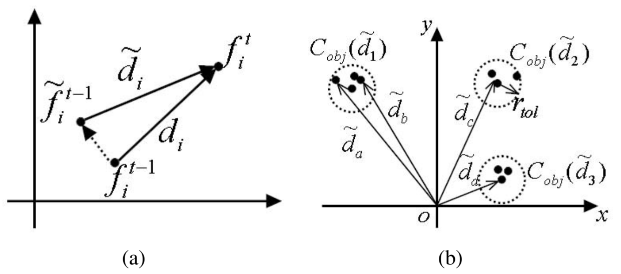


Fig. 2. (a) Visualization of the components of the displacement vector d_i. $\tilde{d}_i$ represents the component related to the real motion of the feature f_i. (b) Clustering on the displacement vector $\tilde{d}_i$. Each cluster has a tolerance radius r_{tol}.

Algorithm 1 Clustering

```
repeat
    Feature extraction and tracking
    Clusters computation
    Background cluster deletion
    Computation of the centre of mass for each cluster
until zooming
```

Cluster computation: To detect the clusters we first need to compute the affine transform for each tracked object. Such a computation is performed over all the features belonging to an object. In particular, let $\hat{A}$ be the computed affine transform, f_i^{t-1} and f_i^t the position of a generic feature respectively at time instant $t-1$ and t. At this point, the effective displacement of the feature i is $\mathbf{d}_i = f_i^t - f_i^{t-1}$, while the estimated one for the effects of the affine transform is as follows:

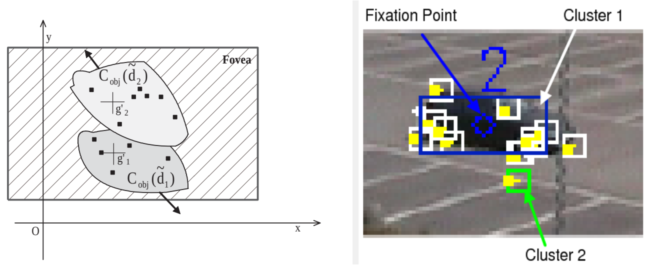
$$\tilde{\mathbf{d}}_i = f_i^t - \tilde{f}_i^{t-1} = f_i^t - \hat{A} f_i^{t-1} \tag{5}$$

where $\tilde{f}_i^{t-1}$ represents the position of the feature i after the compensation of the camera motion by means of the affine transform $\hat{A}$. A graphical representation of the differences among the two displacements, effective and estimated, can be seen in Fig. 2(a). Let TFS_{obj} be the set of features extracted from a window (i.e. *fovea*) centered on the object of interest. Then, the cluster computation is performed respecting the following rule:

$$C_{obj}(\tilde{\mathbf{d}}) = \left\{ f_i \in TFS_{obj} \mid \left\| \tilde{\mathbf{d}}_i - \tilde{\mathbf{d}} \right\|_2 \leq r_{tol} \right\} \tag{6}$$

where $C_{obj}(\tilde{\mathbf{d}})$ is the cluster having all the features i whose displacement $\tilde{\mathbf{d}}_i$ is such that the norm between it and the vector $\tilde{\mathbf{d}}$ is lower than a predefined threshold r_{tol}. In Fig. 2(b) an example of feature clustering is shown.

Background cluster deletion: Once the computation of all the clusters is done, the background cluster can be easily found as well as all the features that have been erroneously extracted from the background in the previous feature extraction step. In particular, after applying the affine transform to the features, if these belong to the background then they should have a null or a small displacement. Hence, to determine the *background*



(a) (b)

Fig. 3. (a) Graphical representation of the computation within the area of interest (e.g. fovea) of the fixation point for each identified cluster. In this way, different objects with different velocities can be tracked even when they cross their trajectories. (b) Computation of two different clusters and of the new fixation point.

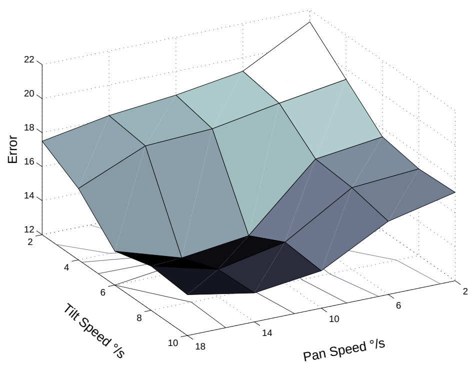


Fig. 5. Localisation error (Euclidean distance expressed in pixels) computed on the sequences classified with respect to the Pan and Tilt rotation speeds.

clusters the rule

$$C_{obj}(\tilde{\mathbf{d}}_k) \in \begin{cases} \text{background} & if \left\| \tilde{\mathbf{d}}_k \right\|_2 \leq r_{tol} \\ \text{object} & otherwise \end{cases} \tag{7}$$

is adopted.

Centre of mass computation: After deleting all the features either belonging to the background or to a cluster with cardinality lower than three (the minimum number of needed features to solve (4)), the technique proposed by Tordoff-Murray [46] can be applied. Thus the fixation point of each cluster is computed. Let $\mathbf{g}_i$ be the fixation point we need to compute for each object i, then we need to solve the following equation:

$$\mathbf{g}'_{\tilde{\mathbf{d}}_k} = N\mathbf{g}_i + \mathbf{r} \tag{8}$$

where $\mathbf{g}'_{\tilde{\mathbf{d}}_k}$ is the new position of the fixation point for the set of features belonging to the cluster $C_{obj}(\tilde{\mathbf{d}}_k)$ and $\mathbf{g}$ is the previous fixation point of the same object. The matrices $N = \begin{pmatrix} n_1 & n_2 \\ n_3 & n_4 \end{pmatrix}$ and $\mathbf{r} = (r_1 \;\; r_2)^T$ are computed using the singular value decomposition SVD on the following linear systems:

$$\begin{pmatrix} f'_1 \\ f'_2 \\ \vdots \\ f'_n \end{pmatrix} (n_1 \; n_2 \; r_1) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{pmatrix} \tag{9}$$

$$\begin{pmatrix} f'_1 \\ f'_2 \\ \vdots \\ f'_n \end{pmatrix} (n_3 \; n_4 \; r_2) = \begin{pmatrix} f_1(y) \\ f_2(y) \\ \vdots \\ f_n(y) \end{pmatrix} \tag{10}$$

where $(f'_1, \ldots, f'_n)^T$ is the vector containing all the features $f'_i = (x, y, 1)$ that are considered well tracked, while $f_i(\cdot)$ is either the x or y coordinate of the feature i at time instant $t-1$. Following this heuristic, a fixation point is computed for each detected cluster which allows to estimate the centre of mass of each moving object, thus avoiding the selection of features in the background or belonging to objects that are not of interest. In Fig. 3(a) a representation of the computation of the fixation point is shown. Fig. 3(b) shows a real example of such a computation. Results for the fixation of rigid and non-rigid objects are presented in Fig. 4.

Since the fixation accuracy is important for the 3D localisation of the objects inside the environment, it is interesting to notice how the fixation error is dependent on the rotation speed of the camera. As matter of fact, in Fig. 5, it can be noticed that the proposed fixation technique works better when the rotation speed increases. This phenomenon can be explained by considering that the method is based on a motion classification technique. In particular, it is based on the capability of segmenting the background from the foreground motion (i.e. background vs. foreground features).

When the camera rotation speed is low, it means that the object is also moving slow and therefore some of the features are misclassified and considered outliers. On the other hand, when the camera rotates faster, the object is also moving faster. Thus the differences of the velocities of the features belonging to the object and to the background increase. This allows to determine more accurate clusters yielding to a better estimation of the fixation points. This aspect is really important when command and control has to be considered for moving the PTZ cameras.

III. Cooperative Camera Networks

Within a visual-surveillance system's architecture, the cooperation of different cameras can occur at different levels. One of these is the tracking level. Recent techniques propose the use of different sensors with overlapping FOVs to achieve a cooperative object tracking and localisation. In [14], Fleuret *et al.* proposed a mathematical framework that allows to combine a robust approach to estimate the probabilities of occupancy of the ground plane at individual time steps with dynamic programming to track people over time. Such a scheme requires that cameras share the majority of their fields of view. While this is affordable for small areas (indoor rooms, alleys, halls, etc.), it becomes infeasible when large environments are considered. In these cases, PTZ cameras can

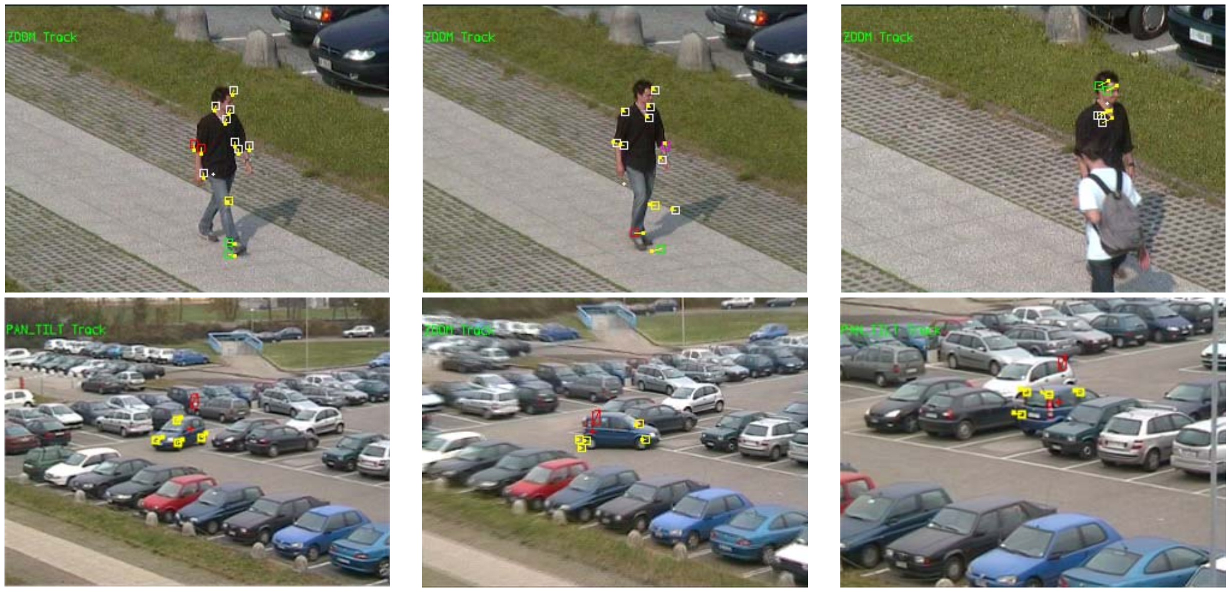


Fig. 4. Sample frames of two test sequences used for evaluating the active tracking of non-rigid and rigid objects. The small boxes represent the tracked feature points organised in clusters depending on their colour. The red-dot represents the computed fixation point on the selected cluster (white for non rigid object and yellow for rigid object).

be exploited to generate two or more overlapping FOVs on areas of interests to solve different problems (e.g., occlusions, tracking and localisation).

A. *Cooperative camera tracking*

When multiple PTZ cameras with different resolution are exploited for tracking, new questions arise [17]. How can standard projective models be extended to the case of heterogeneous and moving cameras? How can the different object resolutions be handled to solve the tracking problems?

In [11], Chen *et al.* derived two different calibration techniques to determine the spatial mapping between a couple of heterogeneous sensors. Such relations can be exploited to improve tracking in a centralized way. Coefficients are assigned—on the basis of the spatial mapping—to the tracking decisions given by different cameras, and the final output follows the highest score to solve ambiguity and occlusions [43].

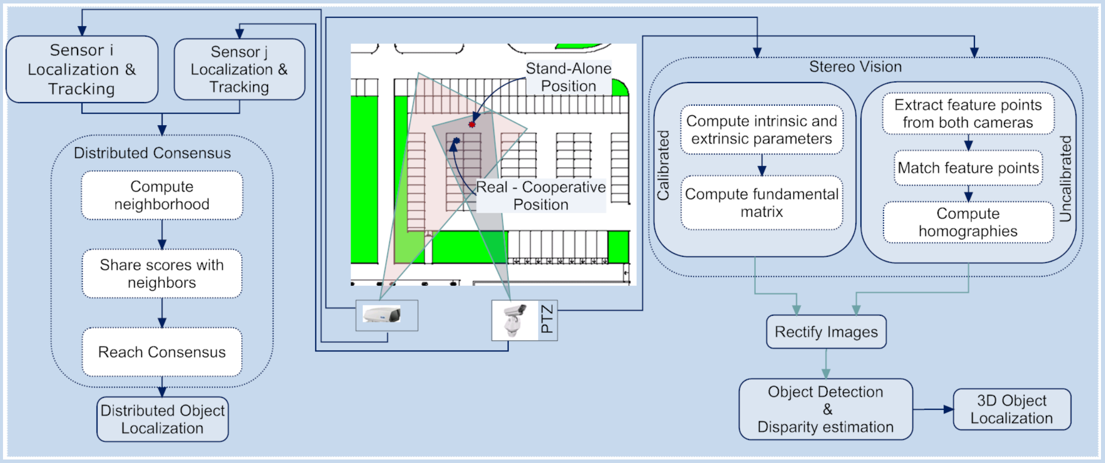
More recently, the distributed approach to the tracking problem has been investigated by exploiting consensus and cooperation in networks [35]. Such a framework requires a consensus (i.e. an agreement regarding a quantity of interest that depends on the state of all agents) achieved with an algorithm that specifies the information exchange among the neighbouring agents. In the context of cooperative camera tracking, the agents are represented by cameras' processes, while the neighbouring relation usually represents the capability of sharing the FOV [45]. Thus, as the target moves through the monitored area, the neighbouring relation defines a dynamic network.

In particular, let a dynamic graph $G = (V, E)$ be the structure adopted to represent the neighbouring nodes/cameras involved in tracking a target. G is modified anytime a camera starts/stops to acquire information on the target or a camera changes PTZ parameters. How to modify G in a proper way is a high-level task. Optimization strategies have to be considered to define the best graph for the required task. Instead, when a graph G is defined, consensus algorithms can be applied to agree on the positions and labels of the targets.

In [45], a Kalman-Consensus tracking is applied. Each camera in the graph determines the ground plane position of each target and computes the information vector and matrix. These are shared together with the target status with the neighbours. A Kalman-Consensus state is computed by each camera for each target by fusing the information received from the neighbours. These distributed approaches allow to reach a distributed estimation of the positions of the targets. Hence, a centralized solution that requires a considerable bandwidth usage from each node to the central unit can be avoided. Anyhow, the localisation accuracy and the tracking performance still depend on the ground plane position estimation performed independently by each camera. Such an estimation is subject to occlusions related errors.

A solution of such a problem can be given by the 3D localization of moving objects by exploiting the cooperativeness between PTZ and static cameras to improve its estimation. In particular, a stereo system of heterogeneous cooperative cameras can be defined to localize moving objects in outdoor areas. Once a target is selected by the surveillance system, a static camera and a PTZ camera can be selected to provide stereo localization of such a target (see Fig. 6).

Calibrated and uncalibrated camera's approaches can be chosen to solve the matching and disparity computation problems. The former are able to analytically determine the calibration matrices using a spherical rectification technique [49] or defining the epipolar geometry of dual heads [18]. The latter [26], even though they require a more complex off-line initialization, do not require the camera calibration and


Sensor i Localization & Tracking
Sensor j Localization & Tracking
Distributed Consensus
Compute neighborhood
Share scores with neighbors
Reach Consensus
Distributed Object Localization
Stand-Alone Position
Real - Cooperative Position
PTZ
Stereo Vision
Calibrated
Compute intrinsic and extrinsic parameters
Compute fundamental matrix
Uncalibrated
Extract feature points from both cameras
Match feature points
Compute homographies
Rectify Images
Object Detection & Disparity estimation
3D Object Localization

Cooperation among cameras is very helpful for solving problems typically present in tracking based on single camera analysis. Cooperation requires first to identify the collaborating cameras and then to decide on the collaboration methods. A centralised approach represents a naive solution in which all the video streams are processed in a dedicated node that commands and controls the entire network. To reduce bandwidth usage of the entire network, distributed approaches can be taken into account. Defining a *neighbouring* property (e.g. cameras having a target in their field of view) allows to cluster cameras in sub-networks. Within and only within each cluster the information (e.g. pre-processed data) is shared between cameras to achieve a common analysis about some traget's property (i.e. position, velocity, etc.). Concerning the cooperative tracking, consensus algorithms appear to be effective in computing the ground-plane position of a target in a distributed way. Anyway, being such distributed techniques dependent to the computations of each node, a real 3D localisation is still subject to occlusions problems. Thus, to have a reliable 3D localisation of a target, stereo vision techniques can be exploited. The system can select the best camera [36] and its closest PTZ camera inside the cluster. Between the two selected image streams, different techniques can be applied to solve the matching problem and the disparity estimation through rectification. For a pair of stereo images $\mathbf{I}_l$ and $\mathbf{I}_r$, the rectification can be expressed as $\mathbf{I}_l^r = \mathbf{H}_l * \mathbf{I}_l \qquad \mathbf{I}_r^r = \mathbf{H}_r * \mathbf{I}_r$ where $(\mathbf{I}_l^r, \mathbf{I}_r^r)$ are the rectified images and $(\mathbf{H}_l, \mathbf{H}_r)$ are the rectification matrices. These rectification transformations can be obtained by minimizing

$$\sum_i [(m_l^i)^T \mathbf{H}_r^T \mathbf{F}_\infty \mathbf{H}_l m_l^i]$$

where (m_l^i, m_r^i) are pairs of matching points between images $\mathbf{I}_l$ and $\mathbf{I}_r$ and $\mathbf{F}_\infty$ is the fundamental matrix for rectified pair of images. To solve the point matching problem, direct methods can exploit SIFT features [29] or edges [31] to look for correspondences within the whole frames. Due to the computational effort of such matching tasks, stereo techniques can be adopted to restrict the search area. These techniques have to take into account the different orientation of the PTZ cameras with respect to the static ones, the different resolution and a possible wide-base line between the two considered stereo sensors.

Fig. 6. Scheme of a cooperative stereo vision system

are more robust when heterogeneous sensors with different resolutions are adopted. In case of uncalibrated approaches, a partial calibration is achieved off-line through a Look-Up-Table (LUT) that is interpolated online to determine the unknown parameters. The LUT contains different pairs of rectification transformations $(\mathbf{H}_r^i, \mathbf{H}_l^i)_{i=1,2\ldots,n}$ for n different pairs of stereo images captured at arbitrary pan and tilt settings.

Construction of the LUT: The main steps to construct the LUT are:

1) Select the different pan angles $p_i|_{i=1,2\ldots m}$ by sampling the whole pan range of the PTZ camera into m equal intervals. The selection of the tilt angular values $t_j|_{j=1,2\ldots n}$ is determined in a similar way.
2) Compute the pairs of rectification transformations $(\mathbf{H}_r^{ij}, \mathbf{H}_l^{ij})_{i=1,2\ldots,m}^{j=1,2\ldots,n}$ for each pair of pan and tilt $(p_i, t_j)_{i=1,2\ldots,m}^{j=1,2\ldots,n}$ values. A rectification algorithm [20] based on the pairs of matching pixels can be used. The pairs of matching pixels are obtained using SIFT matching [29].
3) Store all these $r = m \times n$ pairs of rectification transformations in a LUT.

Interpolation of the LUT: To compute the transformation for every possible configuration of the pan and tilt angles, the LUT values have to be interpolated. The interpolation method receives the pan and tilt angles as input and returns the elements of the corresponding rectification transformations $(\mathbf{H}_r, \mathbf{H}_l)$ as output. This requires to consider a non-linear input-output mapping defined by the function $\mathbf{H} = f(p,t)$. For a known set of input-output values the problem is to find the function $F(.)$ that approximates $f(.)$ over all inputs. That is,

$$\| F(p,t) - f(p,t) \| < \epsilon \qquad \text{for all } (p,t), \tag{11}$$

where ϵ is a small error value. Different non-linear regression tools as Neural-Networks or Support-Vector-Machines can be adopted to solve such a problem.

A further issue is given by image differences due to different view points. Many works on rectification assume that the baseline (the distance between the two cameras) is small if compared to the distance of the object from the cameras, and thus the two images acquired by the cameras are similar. This allows the detection of the matching points using standard techniques such as SIFT matching [29]. However, in an large surveillance system this assumption is no longer valid. It is not rare to have cameras deployed at distances of 50 or more meters one from each other. In this situation, the cooperativeness of two PTZ cameras could give more robust results thank to higher magnifications and the capability of focusing on a specific target. In this context, instead of using wide baseline matching algorithms [31], it is more accurate to use a chain of homographies to extract pairs of matching points [27].

Let $(\mathbf{I}_l^1, \mathbf{I}_r^1)$ be a pair of images of a 3D scene which is far from the cameras along their optical axis. An initial homography $\mathbf{H}^1$ is generated by using extracted pairs of matching points between $\mathbf{I}_l^1$ and $\mathbf{I}_r^1$ using standard feature extractor for wide baseline cases [31]. Let $\mathbf{I}_l^n$ and $\mathbf{I}_r^n$ be a pair of images from the left and the right cameras acquiring a scene/object near to the cameras along their optical axes. The problem is to autonomously extract the pairs of matching points between the images $\mathbf{I}_l^n$ and $\mathbf{I}_r^n$. To solve such a problem, a set of n images is captured from each camera by *virtually* moving the cameras from the initial position (the one at which $(\mathbf{I}_l^1, \mathbf{I}_r^1)$ are acquired) to the current position. Let these two sets of images be $(\mathbf{I}_l^1, \mathbf{I}_l^2 \dots \mathbf{I}_l^n)$ and $(\mathbf{I}_r^1, \mathbf{I}_r^2 \dots \mathbf{I}_r^n)$. The following steps can be adopted:

1) Perform the SIFT matching between image pairs $(\mathbf{I}_l^1, \mathbf{I}_l^2)$, $(\mathbf{I}_l^2, \mathbf{I}_l^3)$, $\dots$, $(\mathbf{I}_l^{n-1}, \mathbf{I}_l^n)$ and use these sets of pairs of matching points for computing their respective homography matrices $\mathbf{H}_l^{1,2}$, $\mathbf{H}_l^{2,3}$, $\dots$, $\mathbf{H}_l^{n-1,n}$.
2) Repeat the procedure given at the above step on the sequence of images captured by the right camera and compute $\mathbf{H}_r^{1,2}$, $\mathbf{H}_r^{2,3}$, $\dots$, $\mathbf{H}_r^{n-1,n}$.
3) Compute the homography matrixes $\mathbf{H}_l$ and $\mathbf{H}_r$
$$\mathbf{H}_l = \prod_{i=0}^{n-2} \mathbf{H}_l^{n-(i+1),n-i} \qquad \mathbf{H}_r = \prod_{i=0}^{n-2} \mathbf{H}_r^{n-(i+1),n-i}$$
4) Compute the homography matrix $\mathbf{H}^n$ for the pairs of matching points between current images $\mathbf{I}_l^n$ and $\mathbf{I}_r^n$ as
$$\mathbf{H}^n = \mathbf{H}_r * \mathbf{H}^1 * (\mathbf{H}_l)^{-1} \tag{12}$$

The cooperation of a PTZ camera with a static camera belonging to the network or among two PTZ cameras results in an improved localisation of the objects in the scene without requiring fixed overlapping fields of view. Such an improvement can be qualitatively seen in Fig. 7. For a quantitative analysis, Fig. 8 represents a surface plot for the localisation error computed on a target with respect to ground truth data. It can be noticed how the stereo localisation keeps the error low. Instead, using a single static or a PTZ calibrated camera, adopting the foot-on-the-ground assumption, the error increases with the distance of the object from the camera and

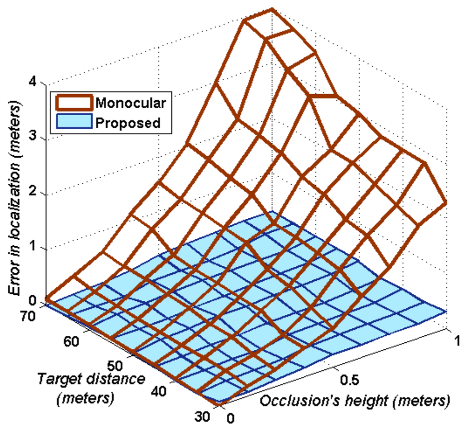


Fig. 8. Error in localisation with respect to the severity of the occlusion (i.e. height) and the object's distance. The proposed cooperative PTZ localisation keeps the error low in all situations while a monocular vision, based on the foot-on-the-ground hypothesis, worsens with the occlusions and distance.

the severity of the occlusion. It is also worth noticing how this problem can be mitigated with centralised or distributed cooperative tracking but not be completely eliminated.

Table I presents a high-level comparison between the three main categorizations for cooperative tracking. The major advantage of centralised approaches are given by receiving all the footages from the cameras, thus allowing to exploit fusion techniques at pixel level without relying on pre-processing results carried out by single sensors. It is clear that such a solution demands a large bandwidth which causes problems for large networks. On the contrary, distributed approaches can reach a consensus by exchanging few data, mainly related to the states of the nodes, within the neighbourhood in charge of tracking a target. In addition, the number of cooperating sensors is limited. This allows the definition of ad-hoc networks to reduce the overall bandwidth usage. However, both approaches suffer from the ground plane constraint in the computation of the target position from single footage.

To overcome this problem, stereo approaches can be considered to reach a more precise 3D localisation. This is achieved by exploiting two sensors thus sidestepping some of the occlusions problems. These sensors have to share a large portion of their points of view (i.e. have a limited baseline) to realize robust matching techniques. This consideration suggests to activate such a technique only when a precise localisation is highly required.

Technique	Advantage	Disadvantage
Centralised	Complete network information Pixel level fusion	High bandwidth usage Affected by occlusions
Distributed	Dynamic network topology Distributed consensus Low bandwidth usage Ad-hoc network communication	Accuracy depending on single camera estimations Affected by occlusions
Stereo	Precise 3D localisation Occlusions free	Constrained points of view Base line constrained

TABLE I
COMPARISON OF COOPERATIVE CAMERA TRACKING APPROACHES.

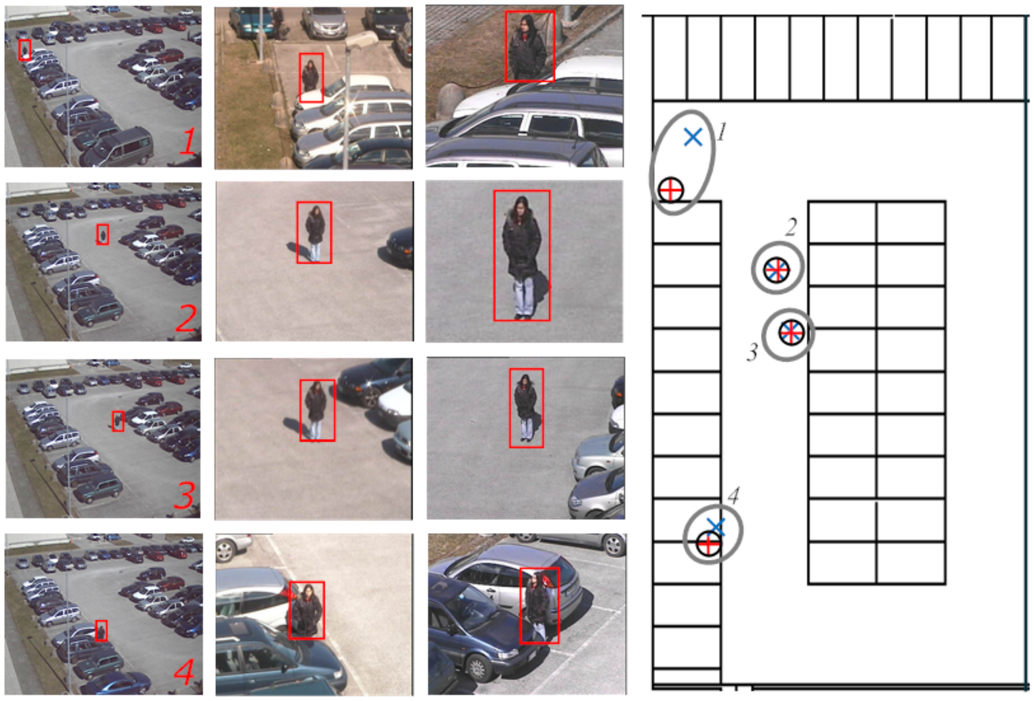


Fig. 7. Example of cooperative PTZ localisation using stereo vision paradigm. Frames from a static camera are shown in the first column, while left and right images of a cooperative PTZs camera are respectively shown in the second and third columns. Two different situations have been considered: a) Occluded person (first and fourth rows) and Non-occluded person (second and third rows). In such situations, the localisation computed by the stereo cooperation between the PTZs (red cross) compared to the one computed by a monocular camera (blue X) is more robust with respect to a ground truth position (black circle). In particular, such an effect is more evident when occlusions occur.

B. PTZ Network Configuration

Another important issue in cooperative camera networks is how the orientations of the cameras can influence the analysis capability of the tracking algorithms. For such a reason, there is a novel research stream that focuses on PTZ network reconfiguration.

Karuppiah *et al.* [22] proposed two new metrics that based on the dynamics of the scene allow to decide the pair of cameras that maximise the detection probability of a moving object. In [21], Kansal *et al.* proposed an optimization process for the determination of the network configuration that maximises the metric. It is interesting to notice how the adopted metrics are concretely bounded to real sensors thus propose a feasible instrument for real applications. More recently, Mittal and Davis [33] introduced a method for determining good sensor configurations that would maximize performance measures for a better system performance. In particular, the authors based the configuration on the presence of random occluding objects and proposed two techniques to analyse the visibility of the objects. Qureshi and Terzopoulos [40] proposed a proactive control of multiple PTZ cameras through a solution that plans assignment and handoff. In particular, the authors cast the problem of controlling multiple cameras as a *multibody* planning problem in which a central planner controls the actions of multiple physical agents. In the context of person tracking, the solution considers the formulation of the relevance $r(c_i, O)$ of a PTZ camera c_i to an observation task O. Five factors are taken into account: a) camera-pedestrian distance r_d, b) frontal viewing direction r_γ, c) PTZ limits $r_{\alpha\beta\theta}$, d) observational range r_o and e) handoff success probability r_h. The planner computes state sequences with the highest probability of success on the basis of a probabilistic objective function. Such a function is given by the probability of success of a state sequence $\mathcal{S}$ over a neighbourhood of cameras N_a and time t as:

$$\mathcal{Q} = \prod_{t\in[0,1,\ldots]} \left(\prod_{i\in[1,N_a]} r(c_i, h_j) \right) \tag{13}$$

where

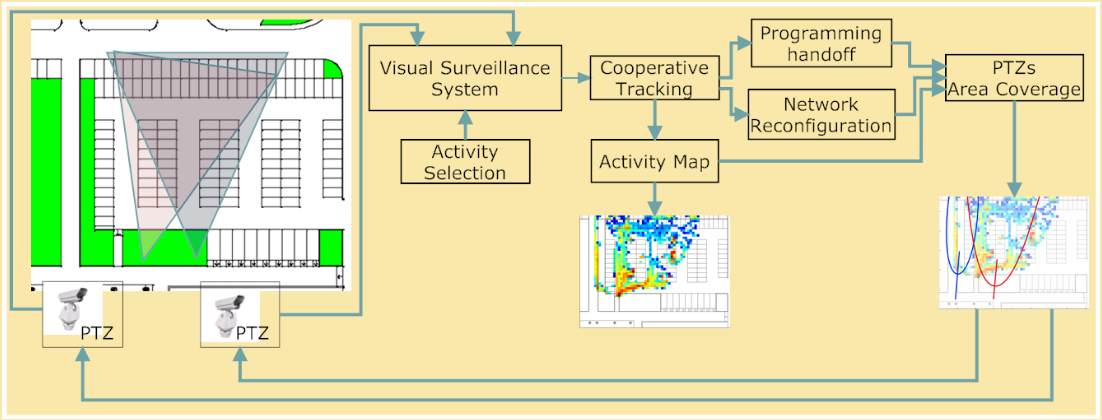
$$r(c_i, h_j) = \begin{cases} 1 & \text{if } c_i \text{ is idle} \\ r_d r_\gamma r_{\alpha\beta\theta} r_o r_h & otherwise \end{cases} \tag{14}$$

The planing is therefore achieved by employing a *greedy best first search* to find the optimal sequence of states.

A different approach to network reconfiguration for person tracking by means of PTZ camera can be developed by employing the game theory. Arslan *et al.* [4], demonstrate that the Nash equilibrium for the strategies lies in the probability distribution. From this formulation, different approaches [28], [44] propose a set of utility functions such as

- Target utility $U_{T_i}(a)$: how well a target T_i is satisfied/acquired while being tracked by some camera;
- Camera utility $U_{C_i}(a)$: how well a camera C_i is tracking a target assigned to it based on user supplied criteria (e.g. size, position, view, etc.) and
- Global utility $U_g(a)$: the overall degree of satisfaction for the tracking performance

and solve the camera assignment by maximizing the global utility function. Different mechanisms to compute the utilities



The goal of the PTZ reconfiguration is to automatically reconfigure the PTZ cameras to improve the system performance. Formally the problem can be seen as a maximization problem $(\mathbf{P}, \mathbf{T}, \mathbf{Z}) = \arg\max_{\mathbf{P},\mathbf{T},\mathbf{Z}} F(\mathbf{P}, \mathbf{T}, \mathbf{Z})$ to select the PTZs configuration parameters that maximise a performance function F of the system. Such a function can be related to a target's property. In such a case, programming the hand-off of the cameras is of primary importance to keep the best acquisition quality. If the function is related to different aspects like targets, cameras and global utilities, a PTZ reconfiguration can be applied to maximize the utility functions. A further interesting problem is to optimally cover the monitored area on the basis of the activity probability. An activity density map can be defined following different strategies such as optimizing tracking performance or detection performance and reducing occlusions. The selected goal has many similarities with 2D data fitting problems, in which the data distribution is approximated by a mixture of density functions (i.e., Gaussians). In our case, there is an activity density map that should be fit by the coverage areas of the PTZ cameras (projection to the ground plane of the cones of view). One of the most popular Mixture-of-Gaussians data fitting algorithms is Expectation-Maximization (EM). In [37], a camera projection model is proposed to project all the activities into compatible camera spaces where the application of the EM algorithm requires less constraints than in the original space. The resulting ellipses determine the PTZ parameters of all the cameras involved in the optimization process.

Fig. 9. Scheme for PTZs reconfiguration

can be provided as in [28], [44], [45], then a bargaining process is executed on the predictions of person utilities at each step. Those cameras with the highest probabilities are used to track the target thus providing a solution to the handoff problem in a video network.

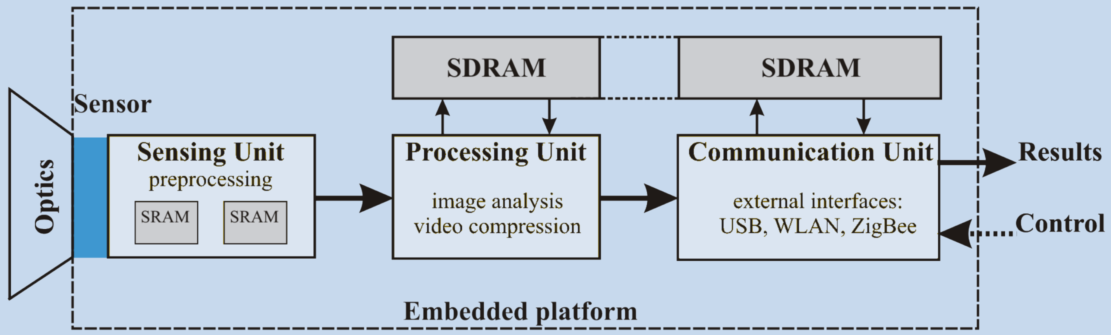
On the other hand, when a PTZ camera is reconfigured to track an object or switched on/off to save power the topology of the network is modified. As consequence, a new configuration is required to provide optimal coverage of the monitored environment. Song *et al.* [44] adopt a uniform distribution of the targets and the coverage resolution utility to negotiate the new network reconfiguration. Piciarelli *et al.* [37] propose a new strategy for the on-line reconfiguration of the network based on the analysis of the activities occurring within the covered area. To achieve the best coverage based on the density of the activities, a constrained Expectation-Maximization (EM) process is introduced to produce the optimal PTZ parameters for a optimal probabilistic coverage of the monitored area. The result is a PTZ network that dynamically can modify its configuration to better acquire the data thus improving the video analysis performance of the network.

IV. Embedded smart cameras

Smart cameras combine image sensing processing and communication on embedded devices [42] (see Fig. 10). Recently, they are deployed in active camera networks [1] where image processing as well as control of the cameras are executed on the embedded nodes. These embedded camera platforms can be classified into single smart cameras, distributed smart cameras and smart camera nodes [41]. The third class of platforms is of special interest, since they combine embedded computation with wireless communication and power-awareness (e.g., [12], [24], [47]).

Integration of image processing algorithms on embedded platforms imposes some challenges because one has to consider architectural issues more closely than on general-purpose computers. Memory is a principle bottleneck for computer system performance. In general-purpose computer systems, caches are used to increase the average performance of the memory system. However, image processing algorithms use huge amounts of data, and often with less frequent reuse. As a result, caches may be less effective. At a minimum, software must be carefully optimized to make best use of the cache; at worst, the memory system must be completely redesigned to provide adequate memory bandwidth [48].

When we have a closer look at low-level signal processing algorithms, we can distinguish five different data access patterns [25]. The way how data is accessed has a great influence on the achievable performance. Data access—and hence the underlying data dependencies—are limiting the degree of parallelization. But parallel execution is a major source of performance gain on dedicated hardware.



An embedded smart camera executes all stages of a typical image processing pipeline onboard. The image sensor converts the incident light of the optics into electrical signals. The sensing unit captures the raw data from the image sensor and performs some preprocessing such as white balance, color transformations and image enhancements. This unit also controls important parameters of the sensor, e.g., frame rate, gain, or exposure, via a dedicated interface. The processing unit reads the preprocessed image data and performs the main image processing tasks and transfers the abstracted data to the communication unit which provides various wired or wireless interfaces such as USB, WLAN or ZigBee.

Low-level operations process the image data at the pixel level but typically offer a high data parallelism. Thus, low-level processing at the sensor unit is often realized on dedicated hardware with fast on-chip memory (SRAM). Dedicated hardware for the sensing unit include application-specific integrated circuits (ASICs), FPGAs [13], or specialized processors [24]. High-level processing at the processing unit, on the other hand, operates on (few) features or objects, which reduces the required data bandwidth but increases the complexity of the operations significantly. Digital signal processors or microprocessors are the prime choice for these tasks. High-level image processing requires much more data storage, thus large off-chip memory (SDRAM) is often attached to the processing and communication units.

Fig. 10. Block diagram of an embedded smart camera.

a) Independent pixel processing: In the simplest case, a single pass over the image is sufficient where the output pixel's value is only dependent on a single input pixel. The processing of each pixel is independent of each other, enabling a full pixel-level data parallel implementation in hardware—even directly on the image sensor. Examples for these full pixel-parallel algorithms include thresholding, color space transformations or simple logic and algorithmic operations.

b) Multi-pass pixel processing: In this class, we consider algorithms which require multiple passes, but the output pixel's value is still only dependent on a single input pixel. Thus, a full pixel-parallel implementation is possible, however synchronization between the multiple passes is necessary. Examples include computation of simple image statistics, histogram equalizations or Hough transforms.

c) Fixed-size block access: Algorithms of this class compute the output pixel's value using multiple input pixels from a fixed block. The data of the input block can typically be accessed in parallel, however for computing the output pixel data dependencies within the block and within neighboring blocks must be considered. Prominent examples of algorithms with such data access pattern include convolution, wavelets and morphological operations.

d) Data-independent global access: Algorithms of this class access multiple source pixels from all over the image to compute the output. Although these algorithms require global access, the access pattern is regular and/or known in advance. Examples include warping or distortion correction.

e) Data-dependent random access: The most complex class of algorithms require access to multiple source pixels from all over the image to compute the output. However, the access pattern is data dependent and therefore not known a priori. Examples of these global image algorithms include flood filling and contour finding.

Algorithm (CIF images)	*time [ms]*	*fps [Hz]*
CAMShift (154 × 154 search block) [39]	2.5	400
Motion detection (single Gaussian) [30]	33	30.0
Motion detection (Kalman) [30]	27	37.0
Adaptive FG/BG modeling [7]	104	9.6

TABLE II
EXECUTION TIMES AND FRAME RATES (CIF RESOLUTION) OF LOW-LEVEL ALGORITHMS ON OUR SMART CAMERA PLATFORM (TMS320C64X DSP @ 600 MHZ).

At first glance, we would expect a linear speedup for the first four access patterns. However, there are several factors limiting the achievable speedup of real hardware implementations such as the memory access times, the available processing elements or some purely sequential control of the algorithms. As consequence, the performance of low-level image processing algorithms may be well below the theoretical speedup. Modern image sensors provide different scanning methods (e.g., windowing, subsampling, random read or binning) to reduce the number of pixel reads resulting in higher frame rates, reduced pixel noise, or reduced power consumption.

In our previous work [8], [9], we developed several embedded smart cameras executing various image processing tasks. Tab. II summarizes the performance of tracking, motion detection and adaptive background/foreground modeling implemented on a TMS320C64x digital signal processor (DSP). The achieved performance of these low-level algorithms—with data access pattern of classes (c), (d) or (e)—on the embedded platform was comparable with reference implementations on standard PC platforms. Baumgartner *et al.* [6] conducted a detailed comparison of low-level image processing algorithms. In most cases, the implementation on PCs and DSPs achieved similar results; the FPGA implementation outperformed the PC and DSP implementations for algorithms with high data parallelism.

Naturally, the native data width and the data path architecture (fixed-point or floating point) influences the implementation on the embedded platform. There is also a tradeoff between the required precision and the power consumption. Fixed-point architectures are more power-efficient; an issue which is getting more and more important.

V. Future smart reconfigurable sensor networks

Future research in the field of smart PTZ camera networks will have to consider the available resources more intensively. Resource-awareness is not only required for economic reasons but also important for scalability, robustness and novel applications. One example for an innovative application is to deploy camera sensor networks in environments with little infrastructure, i.e., the available power supply and communication network will be limited. To provide autonomous operation over some period of time (days or even weeks), the sensor network must use its resources economically. Components and camera nodes will be switched off during idle times and switched on only when required. Thus, the camera network will be dynamically reconfigured to save resources and adapt to some changes in the environment (i.e., switch of cameras at night; wake up nodes when events have been detected in their neighbourhood; avoid PTZ functionality of cameras). Therefore, new optimization algorithms as well as signal processing techniques for fast and computationally efficient video analysis will be needed to support such networks capabilities.

Acknowledgments

This work was funded by the European Regional Development Fund, Interreg IV Italia-Austria Program, Project SRSnet ID 4687.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer Networks*, 51:921–960, 2007.

[2] Y. Aloimonos. *Active Perception*. Lawrence Erlbaum Associates, 1993.

[3] S. Araki, T. Matsuoka, N. Yokoya, and H. Takemura. Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequences. *IEICE Transactions on Information and Systems*, E83-D(7):1583–1591, July 2000.

[4] G. Arslan, J. Marden, and J. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *ASME Journal of Dynamic Systems, Measurement and Control*, 129(5):584–596, 2007.

[5] Y. Bar-Shalom and H. Chen. Multisensor Track-to-Track Association for Tracks with Dependent Errors. *Journal of Advances in Information Fusion*, 1:3–15, 2006.

[6] D. Baumgartner, P. Roessler, W. Kubinger, C. Zinner, and K. Ambrosch. Benchmarks of Low-Level Vision Algorithms for DSP, FPGA, and Mobile PC Processors. In B. Kisacanin, S. S. Bhattacharyya, and S. Chai, editors, *Embedded Computer Vision*, pages 101–120. Springer, 2009.

[7] M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach. Real-Time Video Analysis on an Embedded Smart Camera for Traffic Surveillance. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*, pages 174–181, Toronto, Canada, May 2004.

[8] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed Embedded Smart Cameras for Surveillance Applications. *Computer*, 39(2):68–75, February 2006.

[9] M. Bramberger, R. Pflugfelder, A. Maier, B. Rinner, H. Schwabach, and B. Strobl. A Smart Camera for Traffic Surveillance. In *Proceedings of the First Workshop on Intelligent Solutions for Embedded Systems (WISES 2003)*, Vienna, Austria, June 2003.

[10] A. Cavallaro. Special issue on multi-sensor object detection and tracking. *Signal, Image and Video Processing*, 1, 2007.

[11] C.H. Chen, Y. Yao, D. Page, B. Abidi, A. Koshan, and M. Abidi. Heterogeneous fusion of omnidirectional and PTZ cameras for multiple object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1052–1063, Aug. 2008.

[12] P. Chen, P. Ahammad, C. Boyer, S-I Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Y. Yang, C. Yeo, L.-C. Chang, J.D. Tygar, and S. S. Sastry. Citric: A low-bandwidth wireless camera network platform. In *Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, Stanford, CA, USA, September 2008.

[13] F. Dias, F. Berry, J. Serot, and F. Marmoiton. Hardware Design and Implementation Issues on a FPGA-based Smart Camera. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, pages 20–26, Vienna, Austria, 2007.

[14] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-camera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, February 2008.

[15] G.L. Foresti and C. Micheloni. A robust feature tracker for active surveillance of outdoor scenes. *Electronic Letters on Computer Vision and Image Analysis*, 1(1):21–36, 2003.

[16] G. Guo, C.R. Dyer, and Z. Zang. Linear Combination Representation for Outlier Detection in MotionTracking. In *IEEE International Conference on Computer Vision and Pattern Recognition.*, volume 2, pages 274–281, San Diego, CA, USA, June 20-25 2005.

[17] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Markl, S. Pankanti, A. Senior, C. Fe Shu, and Y.L. Tian. Smart video surveillance. *IEEE Signal Processing Magazine*, 22:38–41, 2005.

[18] J. Hart, B. Scassellati, and S.W. Zucker. Epipolar geometry for humanoid robotic heads. In *Proceedings of the 4th International Cognitive Vision Workshop*, pages 24–36, Santorini, Greece, 12-15 May 2008.

[19] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE Trans. on Pattern Analysis Machine Intelligence*, 20(6):577–589, June 1998.

[20] F. Isgro and E. Trucco. On robust rectification of uncalibrated images. In *International Conference on Image Analysis*, pages 297–302, 1999.

[21] A. Kansal, W. Kaiser, G. Pottie, M. Srivastava, and G. Sukhatme. Reconfiguration methods for mobile sensor networks. *ACM Transaction on Sensor Networks*, 3(4):1–28, 2007.

[22] D. Karuppiah, R. Grupen, A. Hanson, and E. Riseman. Smart resource reconfiguration by exploiting dynamics in perceptual tasks. In *IEEE International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, 2005.

[23] S.M. Khan and M. Shah. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 31(3):505–519, March 2009.

[24] R. P. Kleihorst, A.A. Abbo, B. Schueler, and A. Danilin. Camera mote with a high-performance parallel processor for real-time frame-based video processing. In *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras ICDSC '07*, pages 109–116, 25–28 Sept. 2007.

[25] M. Kölsch and S. Butner. Hardware Considerations for Embedded Vision Systems. In B. Kisacanin, S.S. Bhattacharyya, and S. Chai, editors, *Embedded Computer Vision*, pages 3–26. Springer, 2009.

[26] S. Kumar, C. Micheloni, and G.L. Foresti. *Smart Cameras*, chapter Stereo Vision in Cooperative Cameras Network. N. Belbachir, 2009.

[27] S. Kumar, C. Micheloni, and C. Piciarelli. Stereo localization using dual ptz cameras. In *International Conference on Computer Analysis of Images and Patterns*, volume 5702/2009, pages 1061–1069, Munster, GE, Sep. 2-4 2009.

[28] Y. Li and B. Bhanu. Utility-based dynamic camera assignment and hand-off in a video network. In *IEEE/ACM International Conference*

on Distributed Smart Cameras, pages 1–9, Stanford, USA, 7-11 Sep. 2008.

[29] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–160, 2004.

[30] Martin Mangard. Motion Detection on Embedded Smart Cameras. Master's thesis, Graz University of Technology, 2006.

[31] J. Meltzer and S. Soatto. Edge descriptors for robust wide-baseline correspondence. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, AK U.S.A., June 24-26 2008.

[32] C. Micheloni and G.L. Foresti. Real time image processing for active monitoring of wide areas". *Journal of Visual Communication and Image Representation*, 17:589–604, 2006.

[33] A. Mittal and L.S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76:31–53, 2008.

[34] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(5):449–454, May 1994.

[35] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007.

[36] J. Park, P.C. Bhat, and A.C. Kak. A look-up table approach for solving the camera selection problem in large camera networks. In *Workshop on Distributed Smart Cameras*, pages 72–76, 2006.

[37] C. Piciarelli, C. Micheloni, and G.L. Foresti. Video Network Reconfiguration by Expectation Maximization. In *International Conference on Distributed Smart Cameras*, Como, Italy, 2009.

[38] K.N. Plataniotis and C. S. Regazzoni. Special issue on visual-centric surveillance networks and services. *IEEE Signal Processing Magazine*, 22:12–15, 2005.

[39] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl. Autonomous multicamera tracking on embedded smart cameras. *EURASIP Journal on Embedded Systems Volume 2007*, 10:1–10, 2007.

[40] F.Z. Qureshi and D. Terzopoulos. Planning ahead for PTZ camera assignment and handoff. In *ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8, Como, Italy, Aug-Sep 2009.

[41] B. Rinner, M. Quaritsch, W. Schriebl, T. Winkler, and W. Wolf. The evolution from single to pervasive smart cameras. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, pages 1–10, Stanford, USA, September 7-11 2008.

[42] B. Rinner and W. Wolf. Introduction to distributed smart cameras. *Proceedings of the IEEE*, 96(10):1565–1575, October 2008.

[43] G. Scotti, L. Marcenaro, C. Coelho, F. Selvaggi, and C.S. Regazzoni. Dual camera intelligent sensor for high definition of 360 degrees surveillance". *Visual Image Signal Processing*, 152(2):250–257, Apr. 2005.

[44] B. Song, C. Soto, A. K. Roy-Chowdhury, and J.A. Farrell. Decentralized camera network control using game theory. In *ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8, Stanford, USA, 2008.

[45] C. Soto, B. Song, and A. Roy-Chowdhury. Distributed multi-target tracking in a self-configuring camera network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1486–1493, Miami, USA, 20-25 Jun 2009.

[46] B.J. Tordoff and D.W. Murray. Reactive control of zoom while fixating using perspective and affine cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):98–112, Jan 2004.

[47] T. Winkler and B. Rinner. Pervasive smart camera networks exploiting heterogeneous wireless channels. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 296–299, March 2009.

[48] W. Wolf. *High Performance Embedded Computing*. Morgan Kaufman, San Francisco CA, 2006.

[49] D. Wan J. Zhou. Multiresolution and wide-scope depth estimation using a dual-ptz camera system. *IEEE Transactions on Image Processing*, 18:677–682, 2009.

[50] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.