Abstract—In this paper an approach for dynamic sensor selection in large video-based sensor networks for the purpose of multi-camera object tracking is presented. The sensor selection approach is based on computational geometry algorithms and is able to determine task-relevant cameras (camera cluster) by evaluation of geometrical attributes, given the last observed object position, the sensor configurations and the environment model. Hereby, a special goal of this algorithm is to determine the minimum number of sensors needed to relocate an object, even if the object is temporarily out of sight (e.g., by non-overlapping sensor coverage). It will be shown that the algorithm enables self-organizing tracking approaches to perform optimal camera selection in a highly efficient way. In particular, the approach is applicable to very large camera networks and leads to a highly reduced network and processor load for multi-camera tracking.

I. INTRODUCTION

Video surveillance and monitoring is one of the most recent fields of development and research. Due to the increasing threat by crime, industrial espionage and even terrorism, video surveillance systems became more and more important during the last years.

Especially in large video sensor networks, there is an increasing need for high-class automated surveillance methods. Using a small number of sensors, the operator is able to overlook all videos simultaneously and to switch his focus between them (e.g., to keep a person in view). But by a higher number of sensors the object detection, tracking and camera selection task can no more be done manually by the human operator. In an automated video surveillance system this task relies to be fulfilled by integrated intelligent modules, which provide already processed information about events or location of objects and hereby enable the operator to manage more complex situations.

But, while a lot of research groups focus on multi-camera tracking, only few activities have been noted for intelligent camera selection. However, for very large camera networks the accurate selection of task-related sensors is essential for reduction of bandwidth and computational resources. For this reason, we have developed a novel knowledge-based algorithm, which is able to determine the relevant subset of cameras needed for consistent observation of a single object (or for recognition if the object is temporarily out of sight). Due to the highly efficient camera cluster calculation, the proposed approach is applicable to very large camera networks with overlapping and non-overlapping field-of-views.

This paper is organized as follows: After a short overview of related work in Section II, in Section III the geometric modelling of knowledge about sensors and environment is described, before starting with the introduction to the sensor selection approach in Section IV. In Section V a brief overview of the basic components of our experimental system is provided. In this introduction the integration of the camera selection algorithm in our experimental surveillance system for the purpose of multi-camera human tracking is explained. Finally, the results obtained by evaluations on real and simulated data are shown and future work is presented.

II. RELATED WORK

There are several algorithms in literature for camera selection in sensor networks. Some approaches estimate the observation quality (e.g. quality of detected faces, person velocity in relation to the camera view [1], [2]) or the geometrical relationship between camera and object position ([3], [4]) and select the cameras with the best quality coefficients. In [5] a look-up table is used for camera selection in large networks. Depending on the object position, sensors preassigned to a specific location are selected. In [6] the authors introduce a sensor selection method based on a quality metric called “Appearance Ratio”. The metric characterizes the object detection or segmentation quality of each sensor. Ercan et al. introduce in [7] a sensor selection method based on the minimum MSE of the best linear estimate of the object position as quality metric. The best linear estimate is defined by a chosen camera measurement model.

All these approaches have one thing in common: They are designed to select a subset of sensors out of a set of cameras which all have the object in range. Cameras needed for recognition of the object after disappearance are not considered. Concerning this matter, mainly probabilistic methods have been investigated for learning so-called “transition graphs” of camera networks [8], [9], [10], [11].

The main drawbacks of these methods is the need for a learning period, where correspondences between objects disappearing in one and reappearing in another camera have to be established. Especially, in camera networks with non-static sensors such a learning phase is not applicable in most cases.

Therefore, the camera selection method proposed in this paper is a deterministic knowledge-based approach.
III. KNOWLEDGE AND OBJECT STATE MODELLING

A. Modelling of Camera Sensing Ranges

For each camera or sensor in the network a dedicated video analysis module is assumed (e.g. vision agent on a smart camera). Additionally, self-localization and -calibration capabilities are assumed, which means that each camera is able to provide a polygonal field-of-view (FoV) in world coordinates.

More precisely, for our camera selection approach the so-called visibility polygons are needed. While a FoV represents the maximum observable area defined by the optical attributes of a camera, a visibility polygon is the subset of the FoV, which is directly observable, regarding environmental restrictions (e.g., occlusions by static objects, walls etc.) as well as application-related requirements (e.g., minimum resolution, region-of-interest).

However, by means of the FoV parameters and the prior known building map, it is possible to calculate the visibility polygon. In our system this is done by the visibility algorithm proposed in [12]. As a result of this preprocessing step, for each camera \( S_i \) in the network, a visibility polygon \( L_i = \{l_1, l_2, \ldots, l_{j_i}\} \) is determined (see Fig. 1a).

It is important to mention, that for non-static sensors, the sensing ranges have to be kept up-to-date during processing, to guarantee correct sensor selection.

B. Modelling of Geometrical Prior Knowledge

A central point of our approach is the modelling of both, the environment (building map) and the camera visibility polygons, in a common geometrical 2D representation. The building model is given by a set of \( k \) line segments \( M = \{m_1, m_2, \ldots, m_k\} \) and for each camera \( S_i \in S \) the sensing range is given by a visibility polygon, also represented by a set of line segments \( L_i = \{l_1, l_2, \ldots, l_{j_i}\} \). All line segments are now used to induce a so-called arrangement of line segments \( A(L \cup M) = (E, V, F) \) [13]. That is, \( E \) is a set of edges, \( V \) is a set of vertices and \( F \) is a set of resulting faces in the plane. Hereby, a vertex is the intersection or an end point given a line segment of \( L \cup M \), an edge is the maximum portions of a line without any vertex in between and a face is a subset of the plane that doesn’t contain a point on an edge or a vertex. Consequently, a face is an open polygonal region whose boundary is formed by edges and vertices (Fig. 1c).

For efficient handling of arrangements or planar graphs, the doubly-connected edge list structure (DCEL) turns out to be a standard in computational geometry [13]. Using DCELs, each edge is represented using a pair of directed halfedges, one going from the first endpoint of the line segment to the second one, while the other (its twin halfedge) going in the opposite direction. A DCEL consists of three containers of records: vertices, halfedges and faces, where halfedges are used to separate faces and to connect vertices. That is, from each halfedge the incident faces and vertices are directly on hand. In addition, every halfedge is followed by another halfedge sharing the same incident face, such that the destination vertex of the halfedge is the same as the origin vertex of its following halfedge. Using this structure, it’s possible to traverse the boundary of faces by simply following connected halfedges.

The main advantage is that for DCELs and line arrangements there are highly efficient algorithms for point, edge or face localization and manipulation ([13], [14], [15]). DCEL was chosen, since efficient local manipulation and point localization capability in large arrangements are enhancing the effectiveness of our approach.

C. Object State Model

In this section the object state model used by our sensor selection algorithm is described. For object observation, an arbitrary object detection and tracking approach can be used. Therefore, a generic interface to the sensor selection algorithm was defined.

First of all, we assume the observation process for the object state \( x_k = [x, y]^T \) to be of the form

\[
z_k = x_k + v_k
\]

where \( z_k = [x, y]^T \) is the measurement of the object state \( x_k \) and \( v_k \) represents the measurement uncertainty, that is assumed to be normal distributed with a zero mean and a covariance matrix \( R_k = E[\nu_k \nu_k^T] \).

For camera selection we will introduce two algorithms. For the first one (called basic algorithm, described in Section
IV-A), only the last measured object position \( z_k \) is needed, so the object state estimate is \( \hat{x}_k = z_k \).

However, for the second method (called advanced algorithm, described in Section IV-B), a prediction filter was added as a preprocessing step. The prediction filter is used for temporal adaptation of the state estimate, if no new measurements are available for a longer period of time. Hereby, the predicted process is assumed to be of the form

\[
\hat{x}_k = A\hat{x}_{k-\Delta T} + w_k
\]

where \( A \) is the 2x2 state transition matrix and \( w_k \) represents the process noise, which is assumed to be a normal distributed random variable with a zero mean and a covariance matrix \( Q(\Delta T) = E[w_kw_k^T] \). \( \Delta T \) denotes the elapsed time since the last measurement.

The filtering is performed in two steps: A prediction step, and a update step. The update step simply consists of setting the object state estimate by \( \hat{x}_k = z_k \) and the estimate error covariance \( P_k = R_k \). This is performed asynchronously each time new measurements are available.

The prediction step uses the model parameters \( A \) and \( Q \) to predict the future state estimate \( \hat{x}_k \) and predicted error covariance \( P_k^- \) as follows:

\[
\hat{x}_k^- = A\hat{x}_{k-\Delta T}
\]

\[
P_k^- = AP_k^-A^T + Q(\Delta T)
\]

The prediction step is performed in regular intervals, defined by the process clock of the sensor selection algorithm (e.g., 1Hz). The model parameters \( A \) and \( Q \) of our prediction filter are defined as follows: Since a typical linear motion model (e.g., constant velocity or constant acceleration) is not applicable in nonlinear environments (building map) we use a constant position model with \( I \) as the 2x2 identity matrix (see [16]). In doing so, the predicted state estimate is given by \( \hat{x}_k^- = \hat{x}_{k-\Delta T} = z_{k-\Delta T} \), and is therefore always inside a visibility polygon of a camera. The complete uncertainty about the object movement is modelled as process noise by

\[
Q(\Delta T) = q \begin{pmatrix} \Delta T & 0 \\ 0 & \Delta T \end{pmatrix}
\]

with \( q \) as the assumed position variance (motion dynamic) of the observed object (e.g., 1.5 m/s). Hereby, the error covariance matrix \( Q \) is increasing the state estimate uncertainty, depending on the elapsed time \( \Delta T \) since the last object position measurement. The resulting object state prediction \( \hat{x}^- \) and the predicted error covariance \( P^- \) are now used by the sensor selection approach as information about the object state.

IV. CAMERA SELECTION ALGORITHM / CLUSTERING

In our framework the camera selection algorithm can be considered as an independent process which periodically determines sensors (cluster members) relevant for a certain surveillance task (e.g. tracking). Given \( \hat{x}^- \) and the prior knowledge about sensors and the environment, the camera selection algorithm calculates the relevant subset of the sensor network that is needed for further observation of the object. In addition to determine the sensors with the estimated position in FoV as a first objective, a special goal of this algorithm is to determine the minimum number of sensors needed to relocate an object, even if the object is temporarily out of sight (e.g., by non-overlapping sensor coverage).

For better understanding, we will describe the algorithm in two steps. First, we introduce the camera selection approach given the estimated object position as a single point \( \hat{x}^- \) in 2D space. We will show, that the algorithm in principle is able to determine the relevant sensors for consistent object observation, given only the last valid position estimate.

In a second step, the object state is represented by the object state prediction, including predicted state uncertainty \((\hat{x}^-, P^-)\). We will show how this modification of the algorithm can lead to an optimization of the camera cluster.

A. Basic Algorithm - Single Point Evaluation

The sensors in the camera cluster are split into two groups: One subset of the cluster (the so-called active sensors) are all sensors with the object of interest in view. These sensors, of course, are responsible for object observation. The second subset (so-called passive sensors) are those sensors (with no object in view) needed to limit the local area in the surrounding of the object under observation. The passive sensors are determined for object recognition, in case of object disappearance. Because the sensor selection algorithm guarantees that the object is surrounded by passive or active sensors all time, the observed object has to reappear in one of them.

Now, we will start with our sensor selection method, based on computational geometry algorithms. Let \( S \) be a finite set of sensors, representing the sensor network as introduced in Section V. For each \( S_i \in S \) a set of line segments \( L_i = \{l_1, l_2, \ldots, l_n\} \) (visibility polygon on the 2D ground plane) is given. The collection of the visibility polygons is summarized by \( L = \{L_1, L_2, \ldots, L_m\} \). Additionally, let a set of line segments \( M = \{m_1, m_2, \ldots, m_k\} \) represent the environment model (e.g., building map).

Given this prior information, the sensor selection algorithm is performing the following steps, each time a new object state estimate is provided (see Algorithm 1). The algorithm starts with an initialization step (1). At this time, there is no initial object position available. Thus, \( S_{\text{active}} \) is an empty set. \( S_{\text{passive}} \) includes all sensors in the network, because without position information no spatial containment of the object is possible. Additionally, the complete arrangement of line segments \( A(B) \) is generated, including the line segments of the environment model and all line segments from sensor visibility polygons. Now, in step 2, when an object state estimate is provided (e.g., by a tracking approach), camera selection can be performed. From this time, the algorithm repeats only step 2. First of all, the face \( f_{\text{actual}} \) of the arrangement \( A(B) \) which contains the actual object position is determined. An efficient point location query, as proposed in [17], is used for searching the arrangement cell containing the actual object position. The point location query returns a list of edges, describing the
Fig. 2. The basic algorithm: In a first step (a), the active sensors are determined by a point-in-polygon test. Second, the line segments of active sensors are removed from the arrangement, and the active face (blue area in (b)) is calculated. The sensors which are involved in the face boundary are declared as passive sensors. It is shown, that an object is then surrounded by the camera (c).

face boundary, called connected component of the boundary or CCB in short. While \( f_{\text{active}} \) describes the face boundary including the actual object state, \( f_{\text{active}} \) stands for the face boundary of the previous object state.

Algorithm 1

1) Initialization State (no initial position available)
   a) Add all sensors as passive sensors to the cluster \( S_{\text{active}} = \emptyset, S_{\text{passive}} = S, f_{\text{active}} = \emptyset \) with \( S_{\text{active}} \subseteq S, S_{\text{passive}} \subseteq S \setminus S_{\text{active}} \) and \( f_{\text{active}} \) representing the face in the 2D arrangement of lines, which encloses the object position.
   b) Generate a full arrangement of line segments: \( A(B) = (E, V, F) \) with \( B = L \cup M \)

2) Repeat forever:
   If a new object position estimate \( \hat{x} \) is available, then
   a) Find the face \( f \) in \( A(B) \) which includes the object position: \( f_{\text{actual}} = \{ f_k \in F | \hat{x} \cap f_k \neq \emptyset \} \)
      If \( f_{\text{actual}} = f_{\text{active}} \) then, repeat Step 2, else . . .
   b) Determine the new active sensors:
      \[ S_{\text{active}} = \{ S_i \in S_{\text{cluster}} | \hat{x} \cap \text{Face}(L_i) \neq \emptyset \} \]
      with \( S_{\text{cluster}} = S_{\text{active}} \cup S_{\text{passive}} \) and the \( \text{Face()} \) function representing the infinite set of points in the plane enclosed by the simple polygon \( L_i \).
   c) Determine the passive sensors:
      i) Manipulate the arrangement of line segments: \( A(B') = (E', V', F') \) with \( B' = (L \setminus L_{\text{active}}) \cup M \)
      ii) Find the face \( f' \) in \( A(B') \) which includes the object position
          \( f' = \{ f_k \in F' | \hat{x} \cap f_k \neq \emptyset \} \)
      iii) Determine the sensors involved in the boundary of \( f' \)
          \( S_{\text{passive}} = \{ s_i \in S | L_i \cap \text{CCB}(f') \neq \emptyset \} \)
          where CCB determines the edges which are incident to \( f' \).
   d) \( f_{\text{active}} := f_{\text{actual}} \)

Because recalculation of the camera cluster is necessary if, and only if the object moves to a different face, further steps are only run through if \( f_{\text{actual}} \) differs from \( f_{\text{active}} \).

If a new sensor selection loop is started, the new set of active sensors has to be calculated first (step 2b). Given the object position estimate \( \hat{x} \), for all visibility polygons \( L_i \) of the actual passive and active cameras \( (S_{\text{cluster}}) \) a point-in-polygon-test is performed. Hereby we use an implementation of the well-known ray casting algorithm as described in [15]. The sensors with the object in sensing range are declared as active sensors (Fig. 2a).

It is important to notice, that while for the very first selection loop, all sensors have to be evaluated \( (S_{\text{passive}} = S) \), in subsequent iterations, the point in polygon test is only applied on a highly reduced number of camera polygons and therefore is very effective.

Once the active sensors are known, the passive sensors can be determined in a third step (2c). First, a slightly modified arrangement of line segments \( A(B') \) is created. This is, \( B' \) includes line segments given by the environment model \( M \) and the visibility polygons of all sensors in the network, but without the active ones. The subtraction of the line segments of the active sensors is a crucial part of the algorithm. By excluding the visibility polygons of the active sensors it is guaranteed, that all remaining line segments can be considered as a physical (environment model, e.g., walls) or virtual (entry edges of surrounding cameras) borderline (Fig. 2b).

The arrangement can also be constructed in a very effective way, by temporarily removing the edges which belong to the actual active sensors. In doing so, even in very large sensor networks, the computational power needed for recalculation of the arrangement is minimized.

After generation of the arrangement \( A(B') \), similar as in a previous step, the face \( f' \) (containing the actual object position) is calculated. In a further step, the connected component boundary function CCB determines the edges which are incident to the face \( f' \) (Fig. 2b, blue area). The edges of the CCB can derive from visibility polygons of the sensors \( (L_i) \) or from the environment model \( M \). Because we are only interested in sensor selection, the edges deriving from the building map \( (M) \) are remain unconsidered, and the sensors involved in the CCB are added to \( S_{\text{passive}} \). The sensor

\(^1\text{CCB: connected component of the boundary}\)
Fig. 3. The advanced algorithm involves the object state uncertainty for determination of the probability-of-presence of the object in passive sensors. First the active and tentative passive sensors are calculated as in the basic algorithm (a-c). In further iterations (d-e), the passive sensors are evaluated by the probability-of-presence test. If the probability-of-presence in a passive sensor is higher than a given threshold, then a passive sensor is redefined as active, its line segments are removed from the arrangement and new passive sensors are determined (e). The iterations are repeated until the cluster is stable (f).

selection process is now repeated from Step 2, each time a new object state (position) is available.

We would like to point out, that this algorithm is obviously able to determine locally task-relevant cameras, by recursive determination of cluster members and efficient local manipulation of the arrangement of line segments. However, this approach has two disadvantages: First, the size of the sensor cluster is suboptimal, because no temporal information about object motion dynamics is considered. Second, if an object is able to pass a passive camera without being noticed, a recognition is no more possible, because only cluster sensors are involved in the detection and recognition process, and cluster reorganization in turn, is only triggered by new object observations.

To overcome this drawbacks, a prediction filter as described in Section III-C was added and the algorithm has been slightly modified. In the next section, the advanced algorithm is described.

B. Advanced Algorithm - Involving Predicted Object State/State Uncertainty

The advanced algorithm differs from the basic approach mainly in three aspects: First, as mentioned before, the object state estimate has been substituted by the object state prediction \(\hat{x}^-\), with the temporal adapting uncertainty (error covariance of the predicted state \(P^-\)).

Second, by the given predicted state uncertainty, it is now possible to calculate the probability-of-presence of the object in a passive sensor. If the probability-of-presence in this camera is higher than a given threshold, then the passive sensor is redefined as active (step 2(b)iv). We use the minimum of the Mahalanobis distance \(d_{\text{min}}\) between the object state estimate \(N(\hat{x}^-, P^-)\) and a visibility polygon \(L_i\) as an inverse metric for the probability-of-presence. However, because for calculation of the passive sensors the arrangement of lines has to be modified by removing line segments of all active sensors, the subsequent redeclaration of passive sensors as active implies a recursive sensor cluster calculation. Thus, step 2b of the advanced algorithm is repeated iteratively, until the cluster is stable. That is, until no further passive sensors are subsequently classified as active (step 2(b)iv).

Algorithm 2

1) Initialization state (as in the basic algorithm)
   a) \(S_{\text{active}} = \emptyset, S_{\text{passive}} = S\)
   b) \(A(B) = (E, V, F)\) with \(B = L \cup M\)

2) Repeat forever:
   If a new state prediction \((\hat{x}^-, P^-)\) is available, then
   a) determine the preliminary active sensors:
      \(S_{\text{active}}^- = \{s_i \in S_{\text{cluster}} | \hat{x}^- \cap \text{Face}(L_i) \neq \emptyset\}\)
   b) Repeat until \(S_{\text{active}} = \emptyset\) (cluster stable):
      Determine the tentative passive sensors:
      i) Manipulate the arrangement of line segments:
         \(A(B') = (E', V', F')\) with \(B' = B \setminus L_{\text{active}}\)
      ii) Find the face in \(A(B')\) which includes the predicted object position:
         \(f' = \{f_k \in F' | \hat{x}^- \cap f_k \neq \emptyset\}\)
      iii) Determine the sensors involved in the boundary edges of the face \(f'\):
         \(S_{\text{passive}}^- = \{s_i \in S | L_i \cap \text{CCB}(f') \neq \emptyset\}\)
      iv) Evaluate the min. Mahalanobis dist. \(d_{\text{min}}\) of the tentative passive sensors and redefine them as active, if it’s lower than a given threshold:
         \(S_{\text{active}}^+ = \{s_i \in S_{\text{passive}}^- | d_{\text{min}} < Th\}\) with \(d_{\text{min}} = (N(\hat{x}^-, P^-), L_i)\)
         \(S_{\text{active}} = S_{\text{active}}^- \cap S_{\text{active}}^+\)
         \(S_{\text{passive}} = S_{\text{passive}}^- \setminus S_{\text{active}}^+\)
The third aspect is, that due to the evaluation of the \textit{probability-of-presence} and reclassification of relevant passive sensors as active sensors, it is evident, that only the observation data of active sensors is now needed to be analysed. However, the calculation of passive sensors is still a crucial part of the approach, since it allows for determination of task-relevant sensors by efficient traversal and local manipulation of the arrangement of line segments. The drawback of “losing objects” which passed through the passive sensors unnoticed, is also overcome. If an object remains temporarily undetected, the increasing uncertainty of the predicted object state leads to a successive addition of sensors to the cluster which in turn enables to reorganize the cluster independently from observations. In doing so, even passive sensors have been circumvented, additional sensors are determined first as “preliminary” passive sensors and later as the case may be as active sensors.

V. SYSTEM OVERVIEW

For evaluation of the introduced approach, the camera selection algorithm have been integrated in our experimental video surveillance system. In this section, first an overview of the system architecture is given, and second the integration of the sensor selection approach for multi-camera person tracking is described.

In [18] an architecture, which is based on the idea of an object- and task-oriented data processing, has been proposed (sketched in Fig. 4). By “task-oriented” we mean, that the system focuses on task related events or objects only, and subsequently does not observe all objects in view and does not process all sensors simultaneously. The idea hereby was to design a system, in which autonomous processes (vision agents) are charged with specific surveillance tasks (e.g., tracking of a single object). The vision agents then try to fulfill the dedicated task as well as possible by self-organization and adaptation. In particular, the fusion agents should autonomously determine and process task-relevant sensors only, to minimize network load and sensor data to be analyzed.

The system architecture consists of a structure with one fixed and one dynamic level. The lower level (fixed) summarizes the data sources (sensors) with associated \textit{specialized detection agents} (SDAs). Each sensor is processed by a dedicated SDA, which has the capability of self-localization, motion detection and feature extraction. That is, each SDA (smart camera) is able to determine the position and appearance information of all objects in view and is additionally able to provide information about the camera position, orientation and field-of-view.

The dynamic level consists of temporarily existing tracking agents, the so-called \textit{processing clusters}, or PRCs in short. The PRCs are advanced processing modules, which are identified with a specific surveillance task (e.g. person tracking). A PRC includes two submodules - the so-called \textit{Tracking Module} and the \textit{Dynamic Sensor Manager} (Fig. 4). While the Tracking Module is responsible for data association and fusion of observations provided by cluster sensors, the Dynamic Sensor Manager incorporates the proposed camera selection algorithms for determination of temporarily task-relevant sensors (subscription/unsubscription to SDAs) for further processing. In doing so, the PRCs manage and process multiple sensors (clusters of SDAs) and additionally, are able to autonomously determine those SDAs, needed to fulfill the associated surveillance task (Fig. 4).

For object tracking after starting a new PRC, task-related information (e.g., environment model and initial position of the object to observe) are provided during an initialization step by the human-machine-interface (HMI) or an adequate automatic detection approach. After initialization, there is no further information exchange between the triggering module (e.g., HMI) and the PRC and the Tracking Module is able to track the observed object or to provide a prediction of the object position (as described in III-C).

For evaluation of the approach in very large sensor networks, the same system architecture has been used. Only the SDAs has been replaced by simulated cameras, and simulated tracks.
VI. EXPERIMENTAL RESULTS

A. Evaluation on Real Data

The algorithm has been first evaluated under real conditions, using our experimental surveillance system NEST [19]. Hereby, the person tracking module of the NEST system is designed as described in the section V. The system consists of 20 commercially available IP-cameras installed at our institute. The fields of view of the sensors cover major parts of the test area, but not completely (see Fig. 4(b)). The computational capacity of the evaluation systems consists of 6 off-the-shelf PCs. Four of them temporarily emulate the smart cameras (with 5 running SDAs each). The other processor cores provide computational resources for PRCs (tracking tasks). All 20 SDAs have processed the assigned video streams with 10fps. It is obvious, that due to the distributed preprocessing of video data by the SDAs, the resulting network load is in general negligible (approximately 1KB/sec./observation). In fact, the main problem is not the network load, but the computational resources needed by the Tracking Module of the PRC, if a large number of observations have to be evaluated. For example, if 20 persons are detected by the smart cameras with 10 fps., in the best case approximately 100 observation messages per second are provided asynchronously to the PRC, for data association and fusion. That is, the PRC has to perform data association in less than 5ms, which is hard to achieve. We observed, that processing observations from all 20 sensors in the network leads to dropping messages, even in our relatively small surveillance system. The sensor selection approach in turn, significantly reduces the number of observations to analyze and therefore enables the Tracking Module to perform data association in real time.

For proof of concept, Fig. 5(a) shows the resulting load of the PRC during tracking a single person over several cameras, while other people additionally walked around as clutter. The solid line represents the number of observations from all cameras in the network. The dashed line stands for the number of observation provided by the determined sensor cluster. Fig. 5(b) summarizes the reduction of observations as a percentage of the observations provided by all cameras. We observed, that using the sensor selection approach, it was possible to track the object without dropping any observation message.

B. Evaluations on Simulated Data

The developed simulation tool generates a simple environment model and randomly generates a predefined number of visibility polygons (as described in Section III-A). Fig. 4(c) shows an example of such a configuration, for 100 cameras. The camera network size has been varied between 10 and 500 cameras, with constant density of coverage.

Additionally, a predefined number of object tracks (random walks in the corridors) are generated (green ellipses in Fig. 4(c)). For simulation of different density of traffic, the number of tracks has been varied between 10 and 100. Sensor selection is then applied for single object tracking (as a PRC would do).

Given this framework, we first evaluated the suitability of the camera selection algorithm by comparing the number of observations provided to a tracking approach with and without a Dynamic Sensor Manager. While without a sensor selection approach the number of detection approximately equals the number of objects, our approach reduces the number of observation messages significantly (see Fig. 6(a)). As expected, the plot shows that by a higher density of traffic, the reduction is not as distinctive as in large networks. Nevertheless, compared to the mean number of observation messages (number of tracks), the reduction is quite remarkable (mostly > 90%). Second, to give an overview of the computation complexity, the number of point-in-polygon- and probability-of-presence calculations for different network sizes has been determined (see algorithm description, steps 2a and 2(b)iv). Fig. 6(b) shows, that the point-in-polygon tests are quite stable all the time. This is based on the stable low number of active cameras in the cluster. The probability-of-presence is calculated for all passive sensors and therefore depends on the density and spatial allocation of the cameras. In small sensor networks (e.g., 10 and 25 sensors) almost all sensors are involved in the cluster and therefore the probability-of-presence test is performed for most of the sensors. In larger networks the spatial containment of the sensor cluster shows its advantage and the number of probability-of-presence tests is approximately
constant. Finally, the processing time of our algorithm has been analyzed. Fig. 6 shows the mean runtime for different network sizes, split into four main parts:

- calculation of active sensors (point-in-polygon tests)
- manipulation of the arrangement (removing edges of active sensors)
- calculation of passive sensors (face calculation)
- evaluation of the probability-of-presence + removing edges of new active sensors

It is shown that by increasing the network size, computational costs for sensor selection increase moderately and the approach is applicable to real time systems (absolute runtime ca. 20ms for 500 sensors). However, the passive sensor determination and evaluation takes the major part of the processing time, followed by the task of removing edges of active sensors. This highlights the manipulation of the arrangement (removing lines) as the main cost factor of the algorithm.

VII. FUTURE WORK

In future work, the applicability of our sensor selection approach in non-static camera networks will be studied in detail. The challenge here is, that the computational costs for manipulation of very large arrangements is quite high. However, in our approach, only the updating of the visibility polygons of the temporal cluster sensors is needed. Therefore, we suppose that the additional computational time for updating the arrangement is tolerable and the algorithm is applicable in networks with non-static sensors.

REFERENCES
