3D Target Tracking in Distributed Smart Camera Networks with In-Network Aggregation [*]

Manish Kushwaha
Qualcomm, Inc.
Santa Clara, CA, USA
mkushwah@qualcomm.com

Xenofon Koutsoukos
Institute for Software Integrated Systems (ISIS)
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN, USA
Xenofon.Koutsoukos@vanderbilt.edu

ABSTRACT

With the technology advancements in wireless sensor networks and embedded cameras, distributed smart camera networks are emerging for surveillance applications. Wireless networks, however, introduce bandwidth constraints that need to be considered. Existing approaches for target tracking typically utilize target handover mechanisms between cameras or combine results from 2D trackers into 3D target estimation. Such approaches suffer from scale selection, target rotation, and occlusion, drawbacks associated with 2D tracking. This paper presents an approach for tracking multiple targets in 3D space using a network of smart cameras. The approach employs multi-view histograms to characterize targets in 3D space using color and texture as the visual features. The visual features from each camera, along with the target models are used in a probabilistic tracker to estimate the target state. One of the main innovations in the proposed tracker is in-network aggregation in order to reduce communication cost. The effectiveness of the proposed approach is demonstrates using a camera network deployed in a building.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms

Keywords

Distributed smart cameras, target tracking, in-network aggregation

[*]This work is partially supported by ARO MURI W911NF-06-1-0076.

1. INTRODUCTION

Smart camera networks employ distributed and collaborative algorithms for sensing and information processing [16]. Tracking applications are emerging in the fields of surveillance and industrial vision [12]. Tracking algorithms are often applied in the image plane. These image-plane (or 2D) trackers often run into problems such as target scale selection, target rotation, occlusion, view-dependence, and correspondence across views [17]. There are 3D tracking approaches that fuse results from individual cameras to obtain 3D target trajectories [17, 6]. These approaches employ decision-level fusion, wherein local decisions made by the individual cameras (i.e. 2D tracks) are fused to achieve the global task (i.e. 3D tracks), while discarding the local information (i.e. images captured at the cameras). Because of the decision-level fusion, these approaches also suffer from the problems associated with 2D tracking. The problems inherent to the image-plane based tracking algorithms can be circumvented by employing a tracker in 3D space using a network of smart cameras that can be wireline or wireless. Wireless networks are more appropriate for deployment in complex environments but centralized algorithms may perform poorly due to limited communication bandwidth.

This paper presents an approach for collaborative target tracking in 3D space using a wireless network of smart cameras. The contributions of the paper are threefold: (1) We define a target representation suitable for 3D tracking that includes the target state consisting of the position and orientation of the target in 3D space and the reference model consisting of multi-view feature histograms. (2) We develop a probabilistic 3D tracker based on the target representation and implement the tracker based in sequential Monte Carlo methods using an algorithm that performs in-network aggregation to reduce communication cost. (3) We present a detailed evaluation tracking people in a building which shows robustness against target scale variation and rotation.

A related approach for 3D surveillance using multi-camera system, presented in [7, 1], proposes *Probabilistic Occupancy Map (POM)*, which is a multi-camera generative detection method that estimates ground plane occupancy from multiple background subtraction views. Occupancy probabilities are iteratively estimated by fitting a synthetic model of the background subtraction to the binary foreground motion. Similarly to our work, Bayesian estimation is used. However, there a number of differences between the two approaches. POM performs 2D occupancy estimation on a discretized grid-space, while our tracker estimates 3D position and orientation of a target in 3D space. The POM approach can track dynamic targets because it employs background subtraction for target detection. We can track both static and dynamic targets using **** [Manish, can you complete this sentence?]. The main difference is that our algorithm is designed for resource-constrained wireless networks and employs in-network aggregation for minimizing the communication requirements.

An approach for dynamic 3D scene understanding using a pair of cameras mounted on a moving vehicle is presented in [11]. Scene understanding is a related but different problem from 3D tracking. In 3D scene understanding, targets are recognized and tracked close to the stereo-camera system, which our 3D tracking is targeted for wireless network of smart cameras. The Panoramic Appearance Map (PAM) presented in [8] is similar to our 3D target representation using multi-view histograms. PAM is a compact signature of panoramic appearance information of a target extracted from multiple cameras and accumulated over time. Both PAM and multi-view histograms are discrete representations, and while PAM retains full spatial information, multi-view histograms keep partial spatial information. [Manish, is there any consequence of this difference - full vs. partial?]

The rest of the paper is organized as follows. In Section 2, we briefly discuss some background. In Section 3, we detail the target representation and building blocks for the proposed tracking algorithm, which is presented in Section 4. Section 5 shows performance evaluation results for tracking people moving in a building. We conclude in Section 6.

2. BACKGROUND

The two major components in a typical visual tracking system are the target representation and the tracking algorithm. The target is represented by a reference model in the feature space. Typically, reference target models are obtained by histogramming techniques. For example, the model can be chosen to be the color, texture or edge-orientation histogram of the target. Red-Green-Blue (RGB) colorspace is taken as the feature space in [5], while Hue-Saturation-Value (HSV) colorspace is taken as the feature space in [15] to decouple chromatic information from shading effects.

Consider a target region defined as the set of pixel locations $\{\mathbf{x}_i\}_{i=1\cdots p}$ in an image I. Without loss of generality, consider that the region is centered at $\mathbf{0}$. We define the function $b : \mathbb{R}^2 \rightarrow \{1 \cdots m\}$ that maps the pixel at location $\mathbf{x}_i$ to the index $b(\mathbf{x}_i)$ of its bin in the quantized feature space. Within this region, the target model is defined as $\mathbf{q} = \{q_u\}_{u=1\cdots m}$ with

$$q_u = C \sum_{i=1}^{p} k(\| \mathbf{x}_i \|^2)\delta[b(\mathbf{x}_i) - u] \tag{1}$$

where δ is the Kronecker delta function, C is a normalization constant such that $\sum_{u=1}^{m} q_u = 1$, and $k(x)$ is a weighting function. For example, in [5], this weighting function is an anisotropic kernel, with a convex and monotonic decreasing kernel profile that assigns smaller weights to the pixels farther from the center. If we set $w \equiv 1$, the target model is equivalent to the standard bin counting.

A target candidate is defined similarly to the target model. Consider a target candidate at $\mathbf{y}$ as the region which is a set of pixel locations $\{\mathbf{x}_i\}_{i=1\cdots p}$ centered at $\mathbf{y}$ in the current frame. Using the same weighting function $k(x)$ and feature space mapping function $b(\mathbf{x})$, the target candidate is defined as $\mathbf{p}(\mathbf{y}) = \{p_u(\mathbf{y})\}_{u=1\cdots m}$ with

$$p_u(\mathbf{y}) = C \sum_{i=1}^{p} k(\| \mathbf{y} - \mathbf{x}_i \|^2)\delta[b(\mathbf{x}_i) - u] \tag{2}$$

where C is a constant such that $\sum_{u=1}^{m} p_u(\mathbf{y}) = 1$.

A similarity measure between a target model $\mathbf{q}$ and a target candidate $\mathbf{p}(\mathbf{y})$ plays the role of a data likelihood and its local maxima in the frame indicate the target state estimate. Since both the target model and the target candidate are discrete distributions, the standard similarity function is the Bhattacharya coefficient [4] defined as

$$\rho(\mathbf{y}) \equiv \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] = \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y}) q_u} \tag{3}$$

We use the color model in HSV colorspace developed in [15]. HSV colorspace is approximately uniform in perception. The hue parameter in HSV space represents color information, which is illumination invariant as long as the following two conditions hold, (1) the light source color can be expected to be almost white, and (2) the saturation value of object color is sufficiently large [13]. We also use the texture model based on Local Binary Patterns (LBP) developed in [14]. The most important property of the LBP operator in real-world applications is its tolerance against illumination changes, and its computational simplicity, which makes it possible to analyze images in real-time settings.

3. 3D TARGET TRACKING

In this section, we present the details of the approach. First, we describe the target representation including the target state and the target model and then, we define the similarity measure for localization. We also present an algorithm to estimate the target orientation.

3.1 Target Representation

A target is characterized by a state vector and a reference model. The target state consists of the position, velocity and orientation of the target in 3D space. The reference target model, described below, consists of the 3D shape attributes, and the multi-view histograms of the target object in a suitable feature-space. Such a reference target model would correspond to the actual 3D target which does not change with scale variation and rotation. Once learned during the initialization phase, the model does not need to be updated or learned during tracking.

The state of a target is defined as

$$\boldsymbol{\chi} = [\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}] \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the position, $\mathbf{v} \in \mathbb{R}^3$ is the velocity, and $\boldsymbol{\theta}$ is the orientation of the target in 3D space. Specifically, we represent the target orientation as a unit quaternion [9]. The target orientation can also be represented using Direction Cosine Matrix (DCM), rotation vectors, or Euler angles. Standard conversions between different representations are available. We chose unit quaternions due to their intuitiveness and algebraic simplicity. The target state evolution (the target dynamics) is given by

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{v}_{t-1} \cdot dt + w_{\mathbf{x}} \\ \mathbf{v}_t &= \mathbf{v}_{t-1} + w_{\mathbf{v}} \\ \boldsymbol{\theta}_t &\equiv \boldsymbol{\theta}_{t-1} + w_{\boldsymbol{\theta}} \end{aligned} \tag{5}$$

where $w_{\mathbf{x}}$, $w_{\mathbf{v}}$, and $w_{\boldsymbol{\theta}}$ represent additive noise in target position, velocity, and orientation respectively.

The target model is based on multi-view histograms in different feature-spaces. Since we want to model a 3D target, the definition of target model (see Equation (1)) as a single histogram on an image-plane is not sufficient. We extend the definition of the target model to include multi-view histograms, that is multiple histograms for a number of different viewpoints.

The 3D target is represented by an ellipsoid region in 3D space. Without loss of generality, consider that the target is centered at $\mathbf{x}_0 = [0\ 0\ 0]^{\mathrm{T}}$, and the target axes are aligned with the world coordinate frame. The size of the ellipsoid is represented by the matrix

$$A = \begin{bmatrix} 1/l^2 & 0 & 0 \\ 0 & 1/w^2 & 0 \\ 0 & 0 & 1/h^2 \end{bmatrix} \tag{6}$$

where l, w, h represent the length, width and height of the ellipsoid. A set $\mathcal{S} = \{\mathbf{x}_i\ :\ \mathbf{x}_i^{\mathrm{T}} A \mathbf{x}_i = 1;\ \mathbf{x}_i \in \mathbb{R}^3\}$, is defined as the set of 3D points on the surface of the target. A function $b(\mathbf{x}_i) : \mathcal{S} \rightarrow \{1 \cdots m\}$ maps the surface point at location $\mathbf{x}_i$ to the index $b(\mathbf{x}_i)$ of its bin in the quantized feature space.

Let $\{\hat{\mathbf{e}}_j\}_{j=1\cdots N}$ be the unit vectors pointing away from the target center where N denotes the number of viewpoints. These unit vectors are the viewpoints from where the target is viewed and the reference target model is defined in terms of these viewpoints. The reference target model is defined as

$$\mathbf{Q} = [\mathbf{q}_{\hat{\mathbf{e}}_1}^{\mathrm{T}}, \mathbf{q}_{\hat{\mathbf{e}}_2}^{\mathrm{T}}, \cdots \mathbf{q}_{\hat{\mathbf{e}}_N}^{\mathrm{T}}] \tag{7}$$

where $\mathbf{q}_{\hat{\mathbf{e}}_j}$ is the feature-histogram for viewpoint $\hat{\mathbf{e}}_j$. The feature histogram from viewpoint $\hat{\mathbf{e}}_j$ is defined as $\mathbf{q}_{\hat{\mathbf{e}}_j} = \{q_{\hat{\mathbf{e}}_j,u}\}_{u=1\cdots m}$

$$q_{\hat{\mathbf{e}}_j,u} = C \sum_{\mathbf{x}_i \in \mathcal{R}(\hat{\mathbf{e}}_j)} \kappa\big(d(\mathbf{y}_i)\big)\ \delta[b(\mathbf{x}_i) - u] \tag{8}$$

where δ is the Kronecker delta function, C is a normalization constant such that $\sum_{u=1}^{m} q_{\hat{\mathbf{e}}_j,u} = 1$, $\kappa(\cdot)$ is a weighting function, and

$$\mathcal{R}(\hat{\mathbf{e}}_j) = \{\mathbf{x}_i\ :\ \mathbf{x}_i \in \mathcal{S}, \mathbf{x}_i^{\mathrm{T}} A \hat{\mathbf{e}}_j \geq 0, \tag{9}$$
$$\forall i \neq j \rightarrow \mathbf{y}_i \neq \mathbf{y}_j\} \tag{10}$$

is the set of points on the surface of the target that are visible from the viewpoint $\hat{\mathbf{e}}_j$. In equation (8), $\mathbf{y}_i = P_{\hat{\mathbf{e}}_j} \mathbf{x}_i$ denotes the pixel location corresponding to the point $\mathbf{x}_i$ projected on the image plane, where $P_{\hat{\mathbf{e}}_j}$ is the camera matrix for a hypothetical camera placed on vector $\hat{\mathbf{e}}_j$ with principal axis along $-\hat{\mathbf{e}}_j$. This camera matrix is defined as $P_{\hat{\mathbf{e}}_j} = K\big[R|t\big]$, where $R, \mathbf{t}$ are the rotation and translation given as

$$\begin{aligned} R &= R_{\mathbf{x}}(\theta) R_{\mathbf{y}}(\phi) R_0 \\ \theta &= sin^{-1}(\hat{\mathbf{e}}_{j,z}) \\ \phi &= tan^{-1}\left(\frac{\hat{\mathbf{e}}_{j,y}}{\hat{\mathbf{e}}_{j,x}}\right) \\ R_0 &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \end{aligned}$$

where $R_{\mathbf{x}}(.), R_{\mathbf{y}}(.)$ are the basic rotation matrices along $x-$ and $y-$axis, θ and ϕ are zenith and azimuth angles, respectively, and R_0 is the base rotation. The translation vector $\mathbf{t}$ is given as

$$\begin{aligned} \mathbf{t} &= -R\mathbf{x}_p \\ \mathbf{x}_p &= L\hat{\mathbf{e}}_j \end{aligned}$$

where $\mathbf{x}_p$ is the position of the hypothetical camera places on unit vector $\hat{\mathbf{e}}_j$ at a distance L from the target. The function $d(\mathbf{y}_i)$ in equation (8) computes the pixel distance between pixel locations $\mathbf{y}_i$ and $\mathbf{y}_0$ as

$$d(\mathbf{y}_i) = \big(\mathbf{y}_i - \mathbf{y}_0\big)^{\mathrm{T}} B \big(\mathbf{y}_i - \mathbf{y}_0\big) \tag{11}$$

where $B \in \mathbb{R}^{2\times 2}$ is the representation of size of the ellipse-like shape when the target ellipsoid is projected on the image plane, and $\mathbf{y}_0 = P_{\hat{\mathbf{e}}_j}\mathbf{x}_0$. Finally, the mapping function $b(\mathbf{x}_i)$ maps the 3D point $\mathbf{x}_i$ to the bin in the quantized feature space for pixel at $\mathbf{y}_i = P_{\hat{\mathbf{e}}_j}\mathbf{x}_i$. Since $\mathcal{R}(\hat{\mathbf{e}}_j)$ is the set of points visible from current viewpoint, pixel location $\mathbf{y}_i$ is guaranteed to exist.

3.2 Similarity Measure and Localization

Next, we describe the algorithm to compute the similarity measure between the reference target model and a target candidate state using the camera images from a network of cameras. Consider a camera network of N cameras, where the cameras are denoted as C_n. The camera matrices are denoted as $P_n = K\big[R_n|\mathbf{t}_n\big]$, where K is the internal calibration matrix, R_n is the camera rotation and $\mathbf{t}_n$ is the camera translation. Consider an arbitrary target candidate state $\boldsymbol{\chi} = [\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}]$, and let $\{I_n\}_{n=1\cdots N}$ be the images taken at the cameras at current time-step.

For the target candidate state $\boldsymbol{\chi}$, the *similarity measure* between the target candidate and the reference target model is computed based on the Bhattacharya Coefficient. The similarity measure is defined as

$$\rho(\boldsymbol{\chi}) = \prod_{n=1}^{N} \rho_n(\boldsymbol{\chi}) = \prod_{n=1}^{N} \rho\big(\mathbf{p}_n(\mathbf{x}), \mathbf{q}_{\hat{\mathbf{e}}_n}\big) \tag{12}$$

where N is the number of cameras, $\mathbf{p}_n(\mathbf{x})$ target candidate histogram at $\mathbf{x}$ from camera n, and $\mathbf{q}_{\hat{\mathbf{e}}_n}$ is the target model for the viewpoint $\hat{\mathbf{e}}_n$, where $\hat{\mathbf{e}}_n$ is the viewpoint closest to camera C_n's point-of-view. This is computed as

$$\hat{\mathbf{e}}_n = \arg\max_{\hat{\mathbf{e}}_j}\ \hat{\mathbf{e}}_{\text{target}}^{\mathrm{T}} R(\boldsymbol{\theta}) \hat{\mathbf{e}}_j \tag{13}$$

where $\hat{\mathbf{e}}_{\text{target}}$ is the camera viewpoint toward the target, $\boldsymbol{\theta}$ is the target orientation, and $R(\boldsymbol{\theta})$ is the rotation matrix for the target orientation in terms of the unit quaternion. The unit vector $\hat{\mathbf{e}}_{\text{target}}$ is given by

$$\hat{\mathbf{e}}_{\text{target}} = \frac{\mathbf{x}_n - \mathbf{x}}{\|\ \mathbf{x}_n - \mathbf{x}\ \|} \tag{14}$$

where $\mathbf{x}_n$ is the camera position.

The target candidate histogram $\mathbf{p}_n(\mathbf{x})$ in Equation (12), is computed in a similar way as that for the target model histogram. The target candidate histogram for camera C_n is given by

$$\mathbf{p}_n(\mathbf{x}) = \{p_{n,u}(\mathbf{x})\}_{u=1\cdots m} \tag{15}$$

where

$$p_{n,u}(\mathbf{x}) = C \sum_{\mathbf{y}_i \in \mathcal{R}(\mathbf{x})} \kappa\big(d(\mathbf{y}_i, \mathbf{y})\big)\ \delta[b_I(\mathbf{y}_i) - u] \tag{16}$$

where C is the normalization constant such that $\sum_{u=1}^{m} p_{n,u} = 1$, $\kappa(.)$ is the weighting function, and

$$\mathcal{R}(\mathbf{x}) = \{\mathbf{y}_i\ :\ \mathbf{y}_i \in \mathcal{I}, (\mathbf{y}_i - \mathbf{y})^{\mathrm{T}} B(\mathbf{x})(\mathbf{y}_i - \mathbf{y}) \leq 1, \\ \forall i \neq j \rightarrow \mathbf{y}_i \neq \mathbf{y}_j\}$$

is the set of pixels in the region around $\mathbf{y}$, defined as $B(\mathbf{x})$. Here, $\mathbf{y} = P_n\mathbf{x}$ is the projection of the target position on the camera image plane. The function $d(\mathbf{y}_i, \mathbf{y})$ computes pixel distance between pixel locations $\mathbf{y}_i$ and $\mathbf{y}$ as follows,

$$d(\mathbf{y}_i, \mathbf{y}) = \big(\mathbf{y}_i - \mathbf{y}\big)^{\mathrm{T}} B(\mathbf{x}) \big(\mathbf{y}_i - \mathbf{y}\big) \tag{17}$$

where $B(\mathbf{x}) \in \mathbb{R}^{2\times 2}$ is the representation of the size of the ellipse-like shape when the target ellipsoid is projected on the camera image plane.

3.3 Estimation of Target Orientation

Target orientation is estimated separately from the target position. Below, we describe our algorithm to estimate the target quaternion using the data from multiple cameras. In the first step, we estimate the target quaternion at each camera separately. In the second step, the individual target quaternions are fused together to get a global estimate of the target quaternion.

In the first step, on each camera, we compute the similarity measure of the target candidate histogram, $\mathbf{p}_n(\mathbf{x})$, with each of the histograms in the target reference model (Equation (7))

$$\boldsymbol{\rho}(\boldsymbol{\chi}) \equiv \big[\rho_1(\boldsymbol{\chi}), \rho_2(\boldsymbol{\chi}), \cdots \rho_N(\boldsymbol{\chi})\big] \\ \rho_j(\boldsymbol{\chi}) \equiv \rho\big(\mathbf{p}(\mathbf{x}), \mathbf{q}_{\hat{\mathbf{e}}_j}\big)$$

where $\rho\big(\mathbf{p}(\mathbf{x}), \mathbf{q}_{\hat{\mathbf{e}}_j}\big)$ is the Bhattacharya Coefficient. Now, we have viewpoints $(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \cdots, \hat{\mathbf{e}}_N)$ and similarity measures $(\rho_1, \cdots, \rho_N)$ along each viewpoint. We take the weighted average of all the viewpoints to get the most probable direction of the camera with respect to the target

$$\hat{\mathbf{e}}_{\text{avg}} = \frac{\sum_j \rho_j \hat{\mathbf{e}}_j}{\sum_j \rho_j}$$

The unit vector $-\hat{\mathbf{e}}_{\text{avg}}$ is the estimate of the camera principal axis in the target's frame of reference. To estimate the target rotation vector, we need to compute the transformation between $-\hat{\mathbf{e}}_{avg}$ and $\hat{z}_{\text{CAM}}$, where $\hat{z}_{\text{CAM}}$ is the actual camera principal axis, and apply the same transformation to the target axes, $\mathbf{T} \equiv \mathbf{I}_{3\times 3}$, where $\mathbf{I}_{3\times 3}$ is the identity matrix of size 3.

The transformation between the two unit vectors can be computed as follows,

$$\hat{\mathbf{a}} = \frac{-\hat{\mathbf{e}}_{avg} \times \hat{z}_{\text{CAM}}}{\| \hat{\mathbf{e}}_{avg} \times \hat{z}_{\text{CAM}} \|} \\ \phi = \cos^{-1}\big(-\hat{\mathbf{e}}_{\text{avg}} \cdot \hat{\mathrm{z}}_{\text{CAM}}\big)$$

where $\hat{\mathbf{a}}$ is the Euler axis and ϕ is the rotation angle. Using this transformation, the transformed target axes are

$$T \equiv R_{\hat{\mathbf{a}}}(\phi) = [\hat{\mathbf{e}}_{x'}^{\mathrm{T}}\ \hat{\mathbf{e}}_{y'}^{\mathrm{T}}\ \hat{\mathbf{e}}_{z'}^{\mathrm{T}}] \tag{18}$$

The target orientation on each node is computed using the following conversion from Euler axis and rotation angle to quaternion

$$\hat{\boldsymbol{\theta}}_n = \begin{bmatrix} a_{n,x}\sin(\phi_{\mathrm{n}}/2) \\ a_{n,y}\sin(\phi_{\mathrm{n}}/2) \\ a_{n,z}\sin(\phi_{\mathrm{n}}/2) \\ \cos(\phi_{\mathrm{n}}/2) \end{bmatrix} \tag{19}$$

In the second step, after we have estimated the target quaternions on each of the cameras, we fuse the quaternions together to get a global estimate of the target quaternion. Given target quaternion estimates $\{\hat{\boldsymbol{\theta}}_n\}_{n=1\cdots N}$ and weights $\{w_n\}_{n=1\cdots N}$ from N cameras, we estimate the global target quaternion by taking the weighted average

$$\hat{\boldsymbol{\theta}}_{all} = \frac{\sum_n w_n \hat{\boldsymbol{\theta}}_n}{\| \sum_n w_n \hat{\boldsymbol{\theta}}_n \|}$$

The current target orientation is updated using the global target orientation estimated from the camera images as

$$\hat{\boldsymbol{\theta}} = \alpha \hat{\boldsymbol{\theta}}_{all} + (1-\alpha)\hat{\boldsymbol{\theta}}_{prior}$$

where $\hat{\boldsymbol{\theta}}_{prior}$ is the prior target orientation and α is an update factor.

4. TRACKING ALGORITHM

In visual tracking problems the likelihood is non-linear and often multi-modal. Our tracker is implemented using particle filters. to handle multiple hypotheses and non-linear systems. In a network of wireless cameras, communicating particle weights from each camera to the base station for sensor fusion might be prohibitively expensive. We employ kernel density estimation to compute an approximate density using the particle set on each camera and communicate only the density parameters. The particle density is approximated by using Gaussian Mixture Models (GMMs) and only the mixture model parameters are sent to the base station, thereby reducing the communication cost by a large factor. The communication cost is considerably reduced by performing in-network aggregation on the network routing tree.

4.1 Probabilistic 3D Tracker

At the base station, target state estimation is performed using the GMM parameters received from the cameras. The probabilistic tracker is summarized in Algorithm 1.

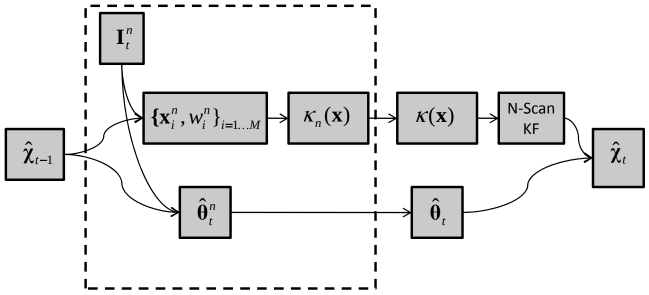
Figure 1 illustrates the tracker operation for a single timestep. At each time step, each camera node performs position estimation and orientation estimation separately. For position estimation, a particle set for the predicted position is generated on each camera node separately.

Afterward, target candidate histograms are computed for each of the proposed particles. In our framework, we use color features, specifically HS colorspace, and texture features, specifically LBP. After we compute the target candidate histograms in HS-space and LBP-space for each particle, we compute the weights according to the following

$$\rho(\mathbf{x}) = \alpha_{\text{HS}}\rho_{\text{HS}}(\mathbf{x}) + (1-\alpha_{\text{HS}})\rho_{\text{LBP}}(\mathbf{x}) \tag{20}$$

Algorithm 1 Probabilistic tracker

```
1: Input: The reference target model $\mathbf{Q}$ (Equation (7)),
   and target state $\hat{\boldsymbol{\chi}}_0 = [\hat{\mathbf{x}}_0, \hat{\mathbf{v}}_0, \hat{\boldsymbol{\theta}}_0]$ at previous time-step.
2:         On Each Camera Node: $\{C_n\}_{n=1\cdots N}$
3:             target position estimation
4:             Generate particle set for the target position,
               $\{\tilde{\mathbf{x}}_i\}_{i=1\cdots M} \sim \mathcal{N}(\hat{\mathbf{x}}_0 + \hat{\mathbf{v}}_0, \Sigma)$
5:             For $i = 1\cdots M$, compute target candidate histogram,
               $\mathbf{p}_n(\tilde{\mathbf{x}}_i)$ according to Equation (15)
6:             For $i = 1\cdots M$, compute weights $w_{n,i} = \rho_n(\tilde{\mathbf{x}}_i)$ according to Equation (12)
7:             Compute 3D kernel density from the particle set
8:             target orientation estimation
9:             Estimate target orientation $\hat{\boldsymbol{\theta}}_n$ according to Equation (19)
10:        On Network Routing Tree:
11:            Aggregate the kernel density from children nodes
12:        On Base Station:
13:            Compute target position estimate, $\hat{\mathbf{x}}$ using mode estimation on the kernel density
14:            Compute target velocity estimate, $\hat{\mathbf{v}}$ using Kalman filter,
15:            Estimate target orientation $\boldsymbol{\theta}$ according to Equation (20).
```



$\mathbf{I}_t^n$
$\{\mathbf{x}_i^n, w_i^n\}_{i=1...M}$
$\kappa_n(\mathbf{x})$
$\kappa(\mathbf{x})$
N-Scan KF
$\hat{\boldsymbol{\chi}}_{t-1}$
$\hat{\boldsymbol{\chi}}_t$
$\hat{\boldsymbol{\theta}}_t^n$
$\hat{\boldsymbol{\theta}}_t$

Figure 1: Probabilistic tracker.

where $\rho_{\text{HS}}(\mathbf{x})$ and $\rho_{\text{LBP}}(\mathbf{x})$ are the similarity measures for the target candidate histograms in HS- and LBP-spaces, respectively (computed using Equation (12)), and $0 \le \alpha_{\text{HS}} \le 1$ is the weighting factor.

Then, the particle set is resampled according to the particle weights, a 3D kernel density is estimated from the resampled particles, and the 3D kernel density is approximated as a 3D-GMM. The kernel density in 3D space is computed as follows

$$\kappa(\mathbf{x}) = \sum_{i=1}^{N} k\left(\frac{\| \mathbf{x} - \mathbf{x}_i \|^2}{h^2}\right)$$

where $k(x) = \exp(-x/2) : [0, \infty) \to \mathbb{R}$ is the kernel profile function. The 3D kernel density $\kappa(\mathbf{x})$ is approximated as a 3D-GMM of appropriate model-order (number of mixture components). A model-order selection algorithm is used to select an optimal model-order that best matches the kernel density. The best matching is done according to KL-divergence as follows

$$m_{opt} = \arg\min_{m \le m_{\text{MAX}}} \text{KL}\big(\kappa(\mathbf{x}) || g_m(\mathbf{x})\big) \tag{21}$$

where $g_m(\mathbf{x}) \equiv \{\alpha_i, \mu_i, \Sigma_i\}_{i=1\cdots m}$ is the 3D-GMM of order m (estimated using the EM algorithm [2]), $\text{KL}(\kappa(\mathbf{x})||g_m(\mathbf{x}))$ is the KL-divergence of $g_m(\mathbf{x})$ from $\kappa(\mathbf{x})$, and m_{opt} is the optimal model-order. Target orientation estimation is performed on each camera node according to the algorithm described in Section 3.

4.2 In-Network Aggregation

Instead of each camera node sending the mixture model parameters to the base station, *in-nodes* in the routing tree aggregate the mixture model parameters and forward a fewer number of parameters. Figure 2 illustrates the in-network aggregation in the tracker. In-network aggregation is done

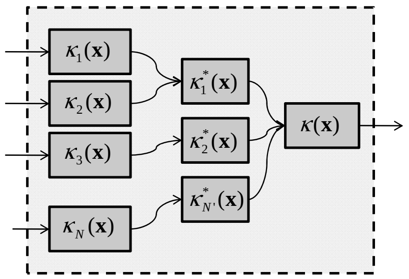


Figure 2: In-network aggregation.

in two-steps. In the first step, the GMMs from multiple camera nodes are combined by taking the product of the GMMs. In the second step, we reduce the number of mixture components in the resulting GMM from the first step to fit the size of a message of fixed size.

Product of GMMs. Let $\{\kappa_j(\mathbf{x})\}_{j=1\cdots N}$ be the kernel densities available at a node from its children and itself. Then, the combined kernel density is given by

$$\kappa(\mathbf{x}) = \kappa_1(\mathbf{x}) \cdot \kappa_2(\mathbf{x}) \cdots \kappa_N(\mathbf{x}) \tag{22}$$

Without loss of generality, we can perform successive pairwise product operations to obtain the combined density. Lets consider the product of two GMMs, $\kappa_1(\mathbf{x})$ and $\kappa_2(\mathbf{x})$ given as

$$\kappa_1(\mathbf{x}) = \sum_{i=1}^{N_1} \alpha_i \mathcal{N}(\mu_i, V_i)$$
$$\kappa_2(\mathbf{x}) = \sum_{j=1}^{N_2} \beta_j \mathcal{N}(\lambda_j, W_j)$$

The product GMM is given by

$$\begin{aligned}\kappa(\mathbf{x}) &= \kappa_1(\mathbf{x})\kappa_2(\mathbf{x}) \\ &= \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} \alpha_i\beta_j \mathcal{N}(\mu_i, V_i)\mathcal{N}(\lambda_j, W_j) \\ &= \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} \gamma_{ij}\mathcal{N}(\xi_{ij}, \Sigma_{ij})\end{aligned}$$

where,

$$\Sigma_{ij} = \left(V_i^{-1} + W_i^{-1}\right)^{-1}$$
$$\xi_{ij} = \Sigma_{ij}\left(V_i^{-1}\mu_i + W_j^{-1}\lambda_j\right)$$
$$\gamma_{ij} = \left[\frac{|\Sigma_{ij}|}{|V_i||W_j|}\right]^{1/2} \frac{\exp(-z_c/2)}{(2\pi)^{D/2}}\alpha_i\beta_j$$
$$z_c = \mu_i^{\mathrm{T}} V_i^{-1}\mu_i + \lambda_j^{\mathrm{T}} W_j^{-1}\lambda_j - \xi_{ij}^{\mathrm{T}}\Sigma_{ij}^{-1}\xi_{ij}$$

So, given two GMMs with N_1 and N_2 mixture components, the product will be a GMM with N_1N_2 mixture components.

Model-Order Reduction. To reduce the number of components in the product GMM such that the mixture model parameters fit a communication message of fixed size is achieved by using a modified k-means algorithm. The model-order reduction problem can be stated as follows. Given a GMM with N components, we want to estimate parameters of a GMM with K components ($K < N$) such that the reduced-order GMM *faithfully* represents the original GMM. In other words,

$$\kappa(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \mathcal{N}(\mu_i, V_i) \equiv \sum_{j=1}^{K} \beta_j \mathcal{N}(\lambda_j, W_j)$$

In the modified k-means algorithm, we want to cluster N points, which are the mixture model components, in K clusters. In the description of the algorithm, we will interchangeably use the terms points and mixture model components. The two key modifications in the standard k-means algorithm are: 1) computation of the distance between points, and 2) the clusterhead update algorithm. First, initialize the k-means algorithm using K random points, $\left(\beta_j^0, \lambda_j^0, W_j^0\right) = (\alpha_i, \mu_i, V_i)$, where $j = 1 \cdots K$ and $i = \textsc{random}(N)$. Then, compute the modified distance of all the points, $i = 1 \cdots N$, with the K clusterheads as

$$d_{ij} = (\mu_i - \lambda_j^0)^{\mathrm{T}}(V_i^{-1} + W_j^{0^{-1}})(\mu_i - \lambda_j^0)$$

and associate each point with the clusterhead closest to it, $m_i = \arg\min_j d_{ij}$, where m_i is the index of the clusterhead closest to the i^{th} point. Then, move the clusterheads to the centroid of the cluster (defined as the collection of all the points associated with the cluster). For $j = 1 \cdots K$, let $\mathcal{C}_j$ represent the set of points in the j^{th} cluster. Then, update the clusterheads according to

$$\beta_j = \sum_{i \in \mathcal{C}_j} \alpha_i$$
$$\lambda_j = \frac{1}{\beta_j}\sum_{i \in \mathcal{C}_j} \alpha_i\mu_i$$
$$W_j = \frac{1}{\beta_j}\sum_{i \in \mathcal{C}_j} \alpha_i(V_i + \mu_i^{\mathrm{T}}\mu_i) - \lambda_j^{\mathrm{T}}\lambda_j$$

The termination criteria is to stop when the clusterheads are converged, $\sum_j \parallel \beta_j - \beta_j^0 \parallel \leq \epsilon$, or the algorithm has exceeded a maximum number of iterations.

Finally, the state estimation is done at the base station by mode estimation on the *combined* kernel density from all the nodes

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \kappa(\mathbf{x})$$

The target position estimate is then used in a *N-scan Kalman smoother* [3] to smooth the position estimates, as well as to estimate the target velocity. Finally, target orientation estimates from each camera node are combined according to Equation (20) to estimate global target orientation.

Communication Cost Analysis. Consider a multihop network of uniform density such that there are n_0 nodes within single-hop. Then, the number of nodes that are k hop away from the center of the network (base-station) is $n_k = (2k-1)n_0$. Hence, the total number of messages transmitted to the base-station for the tracker without in-network aggregation is $M = \sum_{k=1}^{h} kn_k = n_0h(h+1)(4h-1)/6$, where h is the number of hops from the base-station. With in-network aggregation, the number of messages transmitted is, $M_{agg} = \sum_{k=1}^{h} n_k = n_0h^2$. Clearly, the number of messages with in-network aggregation is an order of magnitude smaller ($O(h^2)$) in terms of hops than that without in-network aggregation ($O(h^3)$).

We consider a specific example similar to the network used for evaluating the approach in Section 5. Assume a GMM with 5 components that is stored in a 200 byte structure. For a 2 hop network with $n_0 = 6$ nodes, the number of messages for the tracker without in-network aggregation is 42, whereas for the tracker with in-network aggregation it is 24. If the tracker is running at 4 Hz, then the bandwidth requirements are 33 kbps and 19 kbps respectively for without and with in-network aggregation in a 2 hop network. Similarly, the bandwidth requirements are 105 kbps and 43 kbps respectively in a 3 hop network. In summary, using in-network aggregation, we have approximately 43% less transmissions for a 2 hop network and approx. 59% less for a 3 hop network.

5. PERFORMANCE EVALUATION

In this section, we present results for a real camera network deployment inside our department building (FGH). The setup consists of 6 camera nodes as shown in Figure 3. Figure 3(c) shows the network routing tree for in-network aggregation. We use OpenBrick-E Linux PCs with 533 MHz CPU, 128 MB RAM, a 802.11b wireless adapter, and QuickCam Pro 4000 as video sensors. The QuickCam Pro supports up to 640×480 pixel resolution and up to 30 Hz frame rate. Currently, the cameras record the videos at 4 Hz and 320×240 resolution and tracking is performed offline using a Matlab implementation that realizes the tracker presented in Section 4 following the in-network aggregation scheme. Camera positions are measured manually and camera rotation matrices are estimated using known landmark points in the camera field-of-views. The targets to be tracked are the people moving in the FGH atrium, and target initialization is performed manually at the first time-step. Target model is learned a-priori using a set of images of the target taken from multiple viewpoints. We chose 26 roughly equi-distant viewpoints to cover the target from all sides with sufficient overlap between adjacent views. A detailed evaluation of the trackers as well as several other variations can be found in [10].

Figure 4 shows the camera frames at all six cameras at time-step 1, 40, 80, 120, and 150 during the execution of the tracker. The estimated target positions are shown as blue ellipses superimposed on the camera frames. This experiment demonstrates that even for an extended number of

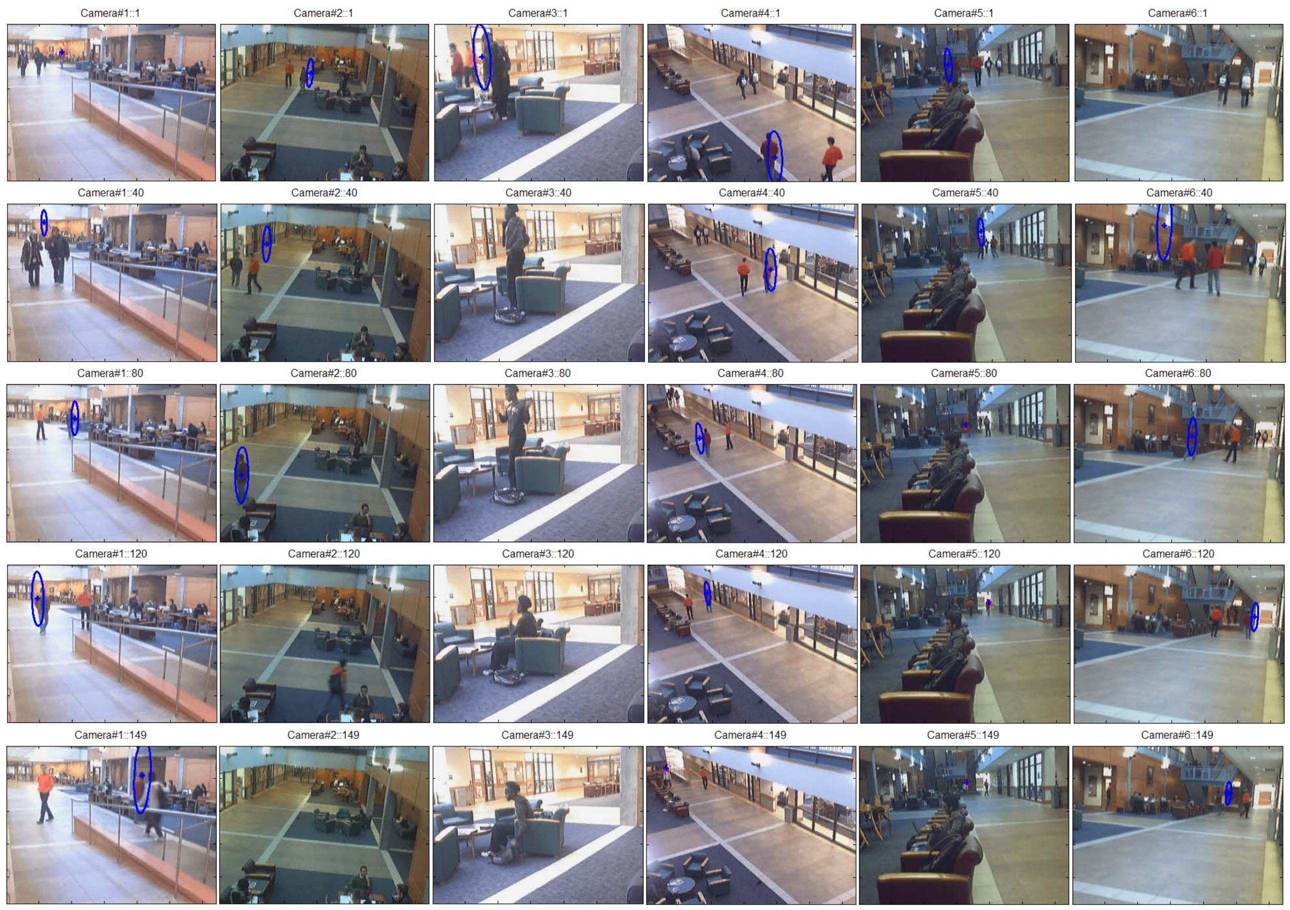


Figure 4: Video target tracking at FGH.

frames (160 frames) the tracker is successfully able to follow the target. During the length of this experiment, the target dramatically changes scale in camera images, comes in and out of different camera fields-of-view, and it is occluded by other people walking in the atrium. This experiment demonstrates the effectiveness of a 3D tracker over state-of-the-art 2D trackers by: 1) not having to learn or update the target model even in case of dramatic scale change and target rotation, and 2) not having to reinitialize a target when it (re-)enters a camera field-of-view. This experiment also demonstrates robustness of the tracker in the presence of target occlusion. Since we are using discriminatory features, i.e. color and texture, and we are performing tracking in 3D space, our tracker is successfully able to handle such occlusions.

Figure 5 shows the 3D target trajectory as estimated by the tracker. Since the sensing region in this setup is large, the target invariably moves in and out of the camera fields-of-view. We have also put a threshold on the size of the projection of the target on the camera image plane. If the pixels occupied by the target in a particular camera image is below the threshold, we deem that frame unusable. Figure 6(a) shows the number of participating cameras at each time-step. At the beginning of the experiment, there are 3 cameras that participate in tracking, which grows to 5 participating cameras in the end. Figure 6(b) shows the percentage of image pixels occupied by the target averaged over the number of participating cameras. At all time-steps, the percentage of image pixels is below 1% of the total pixels. For more tracking results and videos we encourage the reader to go online at `http://tinyurl.com/ya3xqrx` and [10].

6. CONCLUSION

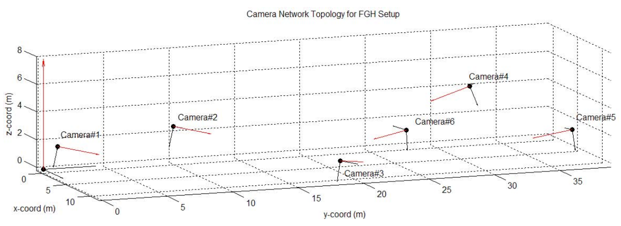
We present an approach for collaborative target tracking in 3D space using a wireless network of smart cameras. We model the targets in 3D space thus circumvent the problems inherent in the tracker based on 2D target models. In addition, we use multiple visual features, specifically, color and texture to model the target. We propose a probabilistic 3D tracker using in-network aggregation, and implemented them using sequential Monte Carlo. We evaluate the tracker for tracking people in a building using a 6-node camera network deployment.

7. REFERENCES

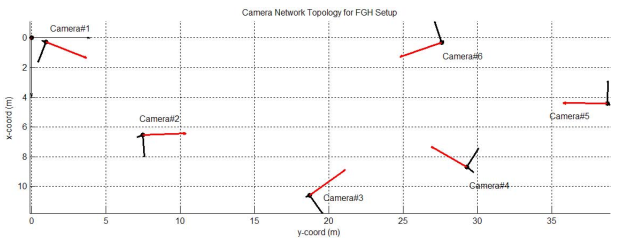
[1] ?? ?? In *??*, 2010.

[2] J. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, ICSI, 1997.

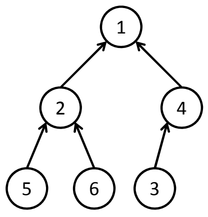
[3] R. G. Brown and P. Y. Hwang. *Introduction to Random Signals and Applied Kalman Filtering with*



(a)



(b)



(c)

Figure 3: Camera network topology – FGH setup (a) 3D-view, (b) top-view, and (c) network routing tree.

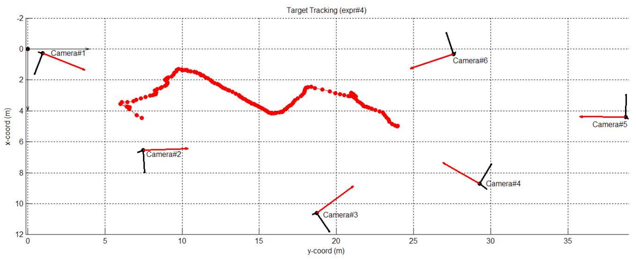


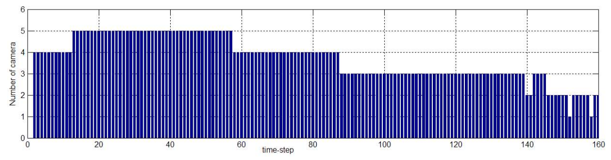
Figure 5: Estimated target trajectory shown with camera network topology.

Matlab Exercises and Solutions, chapter 8: Smoothing. John Wiley & Sons, 3rd edition edition, November 1996.

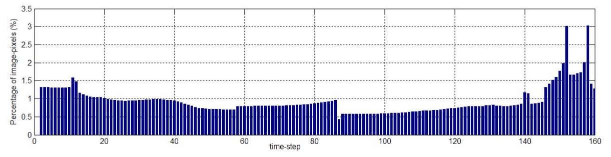
[4] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24, pages 603–619, 2002.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 25, pages 564–577, May 2003.

[6] S. Fleck, F. Busch, and W. Straßer. Adaptive probabilistic tracking embedded in smart cameras for distributed surveillance in a 3d model. In *EURASIP J. Embedded Syst.*, volume 2007, pages 24–24, 2007.



(a)



(b)

Figure 6: Number of *participating* cameras, and average fraction of image pixels occupied by the target.

[7] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. In *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 30, 2008.

[8] T. Gandhi and M. M. Trivedi. Person tracking and reidentification: Introducing panoramic appearance map (pam) for feature representation. volume 18, 2007.

[9] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality.* Princeton University Press, August 2002.

[10] M. Kushwaha. *Feature-Level Information Fusion Methods for Urban Surveillance using Heterogeneous Sensor Networks.* PhD thesis, Vanderbilt University, Nashville, TN, 2010.

[11] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07*, 2007.

[12] C. H. Lin, M. Wolf, X. Koutsoukos, S. Neema, and J. Sztipanovits. System and software architectures of distributed smart cameras. *ACM Trans. Embed. Comput. Syst.*, 9(4):1–30, 2010.

[13] K. Ohba, K. Ikeuchi, and Y. Sato. Appearance-based visual learning and object recognition with illumination invariance. In *Mach. Vision Appl.*, volume 12, pages 189–196, 2000.

[14] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 24, pages 971–987, 2002.

[15] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV '02.*

[16] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf. The evolution from single to pervasive smart cameras. In *ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, 2008.

[17] A. Tyagi, M. Keck, J. Davis, and G. Potamianos. Kernel-based 3d tracking. In *CVPR '07.*