Collaborative Target Tracking using Multiple Visual Features in Smart Camera Networks

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 evolution and fusion of technologies from sensor networks and embedded cameras, smart camera networks are emerging as useful and powerful systems. Wireless networks, however, introduce new constraints of limited bandwidth, computation, and power. Existing camera network approaches for target tracking either utilize target handover mechanisms between cameras, or combine results from 2D trackers into 3D target state for continuous tracking. Such approaches suffer from the drawbacks associated with 2D tracking, such as scale selection, target rotation, and occlusion. In this paper, we present an approach for tracking multiple targets in 3D space using a wireless network of smart cameras. In our approach, we use multi-view histograms in different feature-spaces to characterize targets in 3D space. We employ color and texture as the visual features to model targets. The visual features from each camera, along with the target models are used in a probabilistic tracker to estimate the target state. We demonstrate the effectiveness of our proposed tracker with results tracking people using a camera network deployed in a building.*

1 Introduction

Tracking applications based on distributed and embedded sensor networks are emerging today, both in the fields of surveillance and industrial vision. Tracking based on a single camera cannot provide extended coverage because of limited Field-of-View (FoV). In addition, Single camera 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 [1].

There are a few 3D tracking approaches [1, 2] that fuse results from individual cameras to obtain 3D target trajectories. These approaches employ decision-level fusion, wherein local decisions made by the individual cameras (i.e. 2D tracks) are fused to achieve global decision (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 above-mentioned problems that are inherent in the image-plane based trackers can be circumvented by employing a tracker in 3D space using a network of smart cameras. Such smart camera networks can be employed using wireline or wireless networks. Wireless networks seem more suited due to their easy deployment in complex environments. In wireless networks, traditional centralized approaches have several drawbacks, due to limited communication bandwidth, computational requirements, and thus limiting the spatial camera resolution and the frame rate. We propose an approach for collaborative target tracking in 3D space using a wireless network of smart cameras. The contributions of tthe paper are listed below.

1. We define a target representation, including target state and target reference model, which is suitable for 3D tracking. The target state consists of the position and orientation of the target in 3D space. The target model consists of its multi-view feature histograms. Such a model would correspond to the actual 3D target that does not change, and hence, unlike the image-plane based trackers, the target model does not need to be updated or learned during tracking. We also extend the definition of similarity measure for our proposed target model.

2. We develop a probabilistic 3D tracker based on our new target representation and implement the tracker using sequential Monte Carlo algorithms.

3. We present quantitative evaluation of the tracker using people as targets in a real-world camera network deployment. The results show robustness against target scale variation and rotation, while working within bandwidth constraints.

The rest of the paper is organized as follows. In Section 2, we review the tracking approaches for single camera and camera network systems, and we present details of color and texture features. In Section 3, we detail the target representation and the proposed tracking algorithm. Section 4 shows performance evaluation results for tracking people moving in a building.

2 Background

In this section, we review tracking approaches with a single camera and we describe visual features used for tracking.

2.1 Tracking with Single Camera

The two major components in a typical visual tracking system are the target representation and the tracking algorithm.

2.1.1 Target Representation

The target is characterized by a reference target model in the feature space of interest. Typically, reference target models are obtained by histogramming techniques in the feature space. For example, the model can be chosen to be the color, texture or edge-orientation histogram of the target. In [3], Red-Green-Blue (RGB) colorspace is taken as the feature space, while in [4], Hue-Saturation-Value (HSV) colorspace is taken as the feature space in order to decouple chromatic information from shading effects.

Target Model Consider a target region defined as the set of pixel locations $\{\mathbf{x}_i\}_{i=1\cdots n}$ 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

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

where δ is the Kronecker delta function, C is the normalization constant such that $\sum_{u=1}^{m} q_u = 1$, and $k(x)$ is a weighting function. For example, in [3], 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.

Target Candidate A target candidate is defined similar to the definition of the target model above. Consider a target candidate at $\mathbf{y}$ as the region which is a set of pixel locations $\{\mathbf{x}_i\}_{i=1\cdots n}$ 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

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

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

Similarity Measure A similarity measure between a target model $\mathbf{q}$ and a target candidate $\mathbf{p}(\mathbf{y})$ plays the role of 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 [5] defined as

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

2.1.2 Tracking Algorithm

The two most common approaches for target tracking using a single camera are mean-shift tracking and particle filter-based tracking.

Mean-Shift Tracker In this tracker, the feature histogram-based target representations are regularized by spatial masking with an isotropic kernel [3]. The masking generates spatially-smooth similarity functions. Finally, the target tracking problem is reformulated as finding the local maxima on this spatially-smooth similarity function. The approach in [3] employs a metric derived from the Bhattacharya coefficient as a similarity measure, and uses the mean-shift mode seeking algorithm for maximization.

Particle Filter In the mean-shift tracker, the search for the current estimate is deterministic. A deterministic search might encounter problems when the background close to the target contains similar features, or when the target is partially, or completely occluded momentarily. Unlike the deterministic search in mean-shift tracking, in probabilistic tracking the search is made probabilistically within a sequential Monte Carlo framework [4]. This requires constructing a feature-space likelihood model, which can be based on the similarity measure such as the Bhattacharya coefficient, or any other similarity measure, for example, the Matusita metric [6]. In [4], the color likelihood based on Bhattacharya distance is used with a dynamical state space model for sequential Bayesian estimation using a particle filter.

2.2 Visual Features for Tracking

The most desirable property of a visual feature, also called a visual cue, is its uniqueness and discernibility, so that the targets can be easily distinguished in the feature space. Feature selection is closely related to target representation. For example, color is used as a feature for histogram-based appearance representations, while for contour-based representation, object edges are usually used as features. In complex visual environments, a tracking approach with multiple feature fusion is desirable. The visual features used by our approach are introduced below.

Color In image processing, the RGB colorspace is usually used to represent color. However, the RGB space is not a perceptually uniform color space and is highly sensitive to

illumination changes. HSV colorspace, on the other hand, 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 [7]. In our tracking algorithm, we use the color model in HSV colorspace developed in [4].

Texture Visual textures are the patterns in the intensity variations of a surface. The patterns can be the result of physical surface properties such as roughness, or they could be the result of reflectance differences such as the color on a surface. Image texture is typically defined as a function of the spatial variation in pixel intensities. In our tracking algorithm, we use the texture model based on Local Binary Patterns (LBP) developed in [8]. According to the authors, 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 Probabilistic 3D Tracker

In this section, we present the details of our proposed probabilistic 3D tracker. First, we describe the target representation including the target state and the target model. Then, we define the similarity measure for target localization. We then present an algorithm to estimate target orientation, and finally we present the details of the proposed tracker based on particle filtering.

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.

3.1.1 Target State

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, $\boldsymbol{\theta}$ [9]. 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, albegraic simplicity, and robustness. 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 $w_{\boldsymbol{\theta}}$ are the additive noise in target position, velocity and orientation, respectively.

3.1.2 Target Model

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 multiple histograms for a number of different viewpoints. This is called a multi-view histogram. Our target model is based on multi-view histograms in different feature-spaces.

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. These unit vectors are the viewpoints from where the target is viewed and the reference target model is defined in terms of these viewpoints. Finally, 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 $\hat{\mathbf{e}}_j$, and N is the number of viewpoints. 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 the 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

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

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

$$\mathbf{t} = -R\mathbf{x}_p$$
$$\mathbf{x}_p = L\hat{\mathbf{e}}_j$$

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 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 $\mathbf{y}_0 = P_{\hat{\mathbf{e}}_j}\mathbf{x}_0$.

3.1.3 Similarity Measure and Localization

Below, 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

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

where $\hat{\mathbf{e}}_{\mathrm{target}}$ is the camera viewpoint towards 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 $\hat{\mathbf{e}}_{\mathrm{target}}$ is given by

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

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.

3.1.4 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, $\mathbf{T} \equiv \mathbf{I}_{3\times3}$, where $\mathbf{I}_{3\times3}$ is the identity matrix of size 3.

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

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

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

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

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

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

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.

3.2 Tracking Algorithm

In visual tracking problems the likelihood is non-linear and often multi-modal. As a result linear filters such as the Kalman filter, and its approximations are usually not suitable. Our tracker is implemented using particle filters. Particle filters can 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. At the base station, target state estimation is performed using GMM parameters received from the cameras. The probabilistic tracker is summarized in Algorithm 1.

Algorithm 1 Probabilistic tracker

1: Input: The reference target model $\mathbf{Q}$ (Equation (7)), and target state $\hat{\boldsymbol{\chi}}_0 = [\hat{\mathbf{x}}_0, \hat{\mathbf{v}}_0, \hat{\boldsymbol{\theta}}_0]$ in previous time-step.
2: ON EACH CAMERA NODE: $\{C_n\}_{n=1\cdots N}$
3: TARGET POSITION ESTIMATION
4: Generate *synchronized* 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: TARGET ORIENTATION ESTIMATION
8: Estimate target orientation $\hat{\boldsymbol{\theta}}_n$ according to Equation (19)
9: ON BASE STATION:
10: For $i = 1\cdots M$, combine weights from each camera, $w_i = \prod_n w_{n,i}$,
11: Compute target position estimate, $\hat{\mathbf{x}}$
12: Compute target velocity estimate, $\hat{\mathbf{v}}$ using Kalman filter,
13: Estimate target orientation $\boldsymbol{\theta}$ according to Equation (20).

Figure 1 illustrates the tracker operation for a single time-step. 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. Afterwards, 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

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

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 $0 \leq \alpha_{\text{HS}} \leq 1$ 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

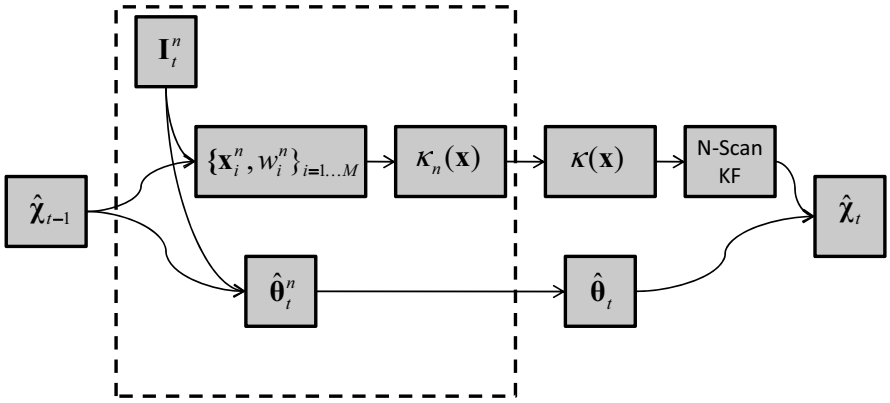


Figure 1: Proposed tracker for video tracking.

follows

$$\kappa(\mathbf{x}) = \sum_{i=1}^{N} k\left(\frac{\| \mathbf{x} - \mathbf{x}_i \|^2}{h^2}\right) \tag{21}$$

where $k(x) = \exp(-x/2) : [0, \infty) \rightarrow \mathbb{R}$ 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

$$m_{opt} = \arg \min_{m \leq m_{\text{MAX}}} \text{KL}\big(\kappa(\mathbf{x}) || g_m(\mathbf{x})\big) \tag{22}$$

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 [10]), $\text{KL}(\kappa(\mathbf{x})||g_m(\mathbf{x}))$ 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 earlier in the section.

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

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \kappa(\mathbf{x}) \equiv \arg\max_{\mathbf{x}} \sum_{i=1}^{m_{opt}} \alpha_i \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{23}$$

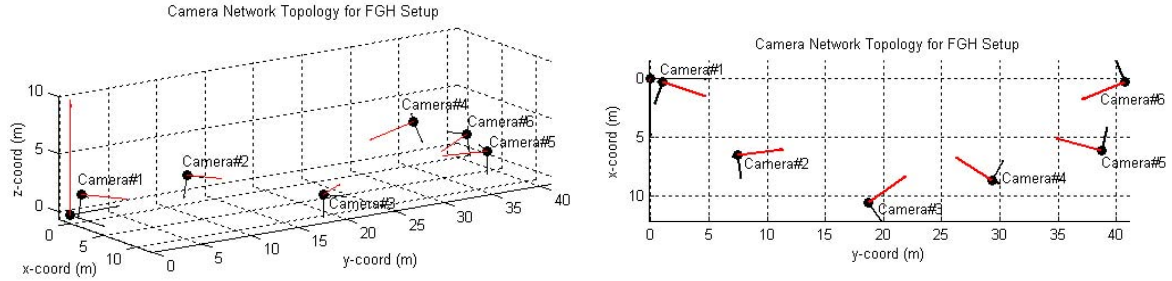
The target position estimate is then used in a *N-scan Kalman smoother* [11] to smooth the position estimates, as well as to estimate the target velocity. Finally, target orientation estimates from each camera node are combined according to Equation (20) to estimate global target orientation.

4 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 2. The targets to be tracked are the people moving in the FGH atrium. Currently, the cameras store the videos and tracking is performed offline using a Matlab implementation that realizes the tracker presented in Section 3. The algorithm performs 3D tracking that incurs high computational cost but employes 3D kernel density estimate using a GMM that reduces the communication requirements. A detailed evaluation of the tracker as well as several variations including a detailed computational complexity analysis can be found in [12].

Figure 3 shows the camera frames at all six cameras at time-step 1, 31, 50, 75, 100, 150 and 175 during the execution of the tracker. Target initialization is performed manually at the first time-step. The estimated target positions are shown as blue ellipses superimposed on the camera frames. This experiment demonstrates that even for an extended number of frames (192 frames) the tracker is successfully able to follow the target. During the length of this experiment, the target dramatically changes scale in camera images, and comes in and out of different camera fields-of-view. 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 3) not having to reinitialize a target when it (re-)enters a camera field-of-view.

Figure 4 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 5(a) shows the number of participating cameras at each time-step. At the beginning of the experiment, there are 2 cameras that participate in tracking. It grows up to six cameras for a single time-step before dropping to three participating cameras in the end. At all time-steps, there are at least 3 cameras that contain the target in their fields-of-view, such that the camera projection size exceeds the threshold. Figure 5(b) shows the percentage of image pixels occupied by the target averaged over the number of participating cameras. As we can see, for most of the time the



(a) (b)

Figure 2: Camera network topology – FGH setup (a) 3D-view, and (b) top-view.

Figure 3: Video target tracking at FGH.

percentage of image pixels is below 1% of the total pixels. The share of pixels for the target grows above 1% when it is in the middle of the sesing region. For more tracking results and videos we encourage the reader to go online at `http://tinyurl.com/ya3xqrx`.

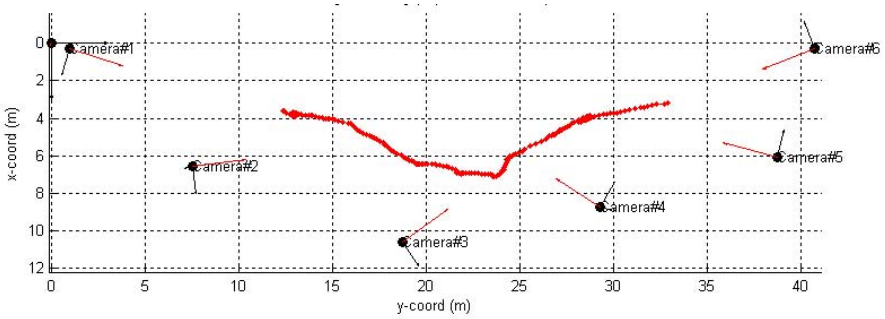
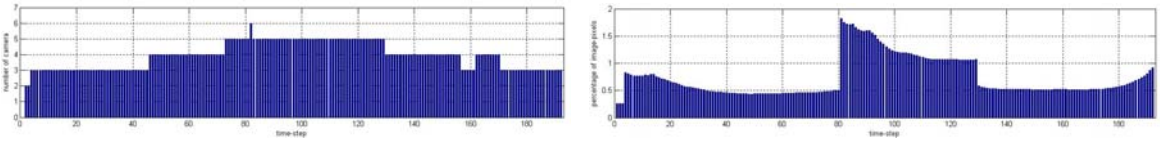


Figure 4: Estimated target trajectory shown with camera network topology.



(a) (b)

Figure 5: Number of *participating* cameras, and average fraction of image pixels occupied by the target.

5 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 implemented using sequential Monte Carlo. We evaluate the tracker for tracking people in a building using a 6-node camera network deployment.

References

[1] A. Tyagi, M. Keck, J. Davis, and G. Potamianos, "Kernel-based 3d tracking," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[2] 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.*, vol. 2007, no. 1, 2007, pp. 24–24.

[3] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, May 2003, pp. 564–577.

[4] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, 2002, pp. 661–675.

[5] 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*, vol. 24, 2002, pp. 603–619.

[6] K. Matusita, "Decision rules, based on the distance, for problems of fit, two samples, and estimation," vol. 26, no. 4, 1955, pp. 631–640.

[7] K. Ohba, K. Ikeuchi, and Y. Sato, "Appearance-based visual learning and object recognition with illumination invariance," in *Mach. Vision Appl.*, vol. 12, no. 4, 2000, pp. 189–196.

[8] 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.*, vol. 24, no. 7, 2002, pp. 971–987.

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

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

[11] R. G. Brown and P. Y. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions*, 3rd ed. John Wiley & Sons, November 1996, ch. 8: Smoothing.

[12] M. Kushwaha, "Feature-level information fusion methods for urban surveillance using heterogeneous sensor networks." Ph.D. dissertation, The Vanderbilt University, Nashville, TN, 2010.