Adaptive Fuzzy Particle Filter Tracker for a PTZ Camera in an IP Surveillance System

Parisa Darvish Zadeh Varcheie and Guillaume-Alexandre Bilodeau
Department of Computer Engineering and Software Engineering, École Polytechnique de Montréal
P.O. Box 6079, Station Centre-ville, Montréal, (Québec), Canada, H3C 3A7
{parisa.darvish-zadeh-varcheie, guillaume-alexandre.bilodeau}@polymtl.ca

Abstract

We propose an adaptive fuzzy particle filter (AFPF) method adapted to general object tracking with an IP PTZ camera. Particle filter samples are weighted using fuzzy membership functions and are applied to geometric and appearance features. In our particle filter, targets are modeled and tracked based on sampling around predicted positions obtained by a position predictor and moving regions detected by optical flow. Sample features are scored based on fuzzy rules. In this paper, we apply the AFPF to a human tracking application in an IP PTZ surveillance system. Results show that our system has good target detection precision ($> 93.9\%$), low track fragmentation, and a high processing rate, and that the target is almost always located within $1/6^{th}$ of the image diameter from the image center.

Index Terms

People tracking, low frame rate tracking, IP PTZ camera, particle filter, fuzzy logic.

I. Introduction

Object tracking is an active subject and has many applications in the field of computer vision. A challenging problem in video surveillance is how to identify and recognize currently occurring events. Accurate, efficient, and reliable tracking is required for this task. Here, object tracking is applied to human upper body, which determines the location of the upper body for each input image of a video. It can be used to obtain images of the face of a human target in different poses. We track the upper body (face and torso), because faces seen from afar do not have discriminative features, other than skin color, and might not be visible from some viewpoints. Thus, torso information that includes additional colors improves tracking accuracy in different scales and from different viewpoints. We propose to track the human upper body by means of an IP PTZ camera (a network-based camera that pans, tilts, and zooms). An IP PTZ camera responds to commands via its integrated Web server, allowing a distributed access from the Internet (access from everywhere, but with a non defined delay). Tracking with such a camera implies:

- an irregular response time to control commands;

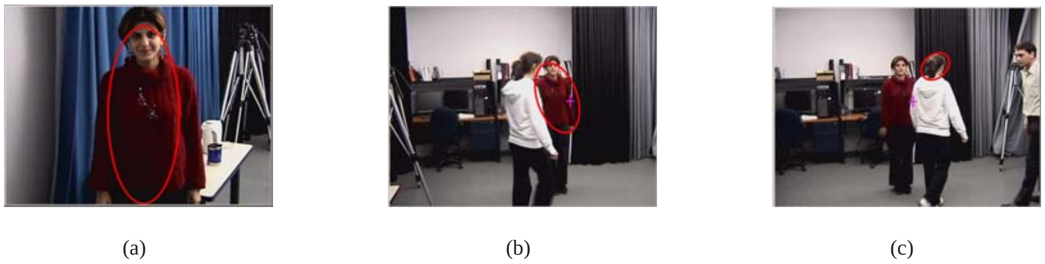


Fig. 1. CamShift tracking failure. Track with CamShift (red ellipse) and track with our proposed method (pink cross) for frames of E_6, (a) initial model selection for CamShift (b) and (c) short-term occlusion and camera panning.

- a low frame rate (while the camera executes the motion command, the frames received are mostly useless, as no new command can be sent without a delay);
- an irregular frame rate, because of network delays (the time between two frames is not necessarily constant);
- a changing field of view (FOV), resulting from panning, tilting, and zooming;
- various scales of objects.

To illustrate some of the problems we face with our network camera, we applied the CamShift method for tracking the face of a person. Fig. 1 shows a tracking failure for two consecutive frames, which is the result of processing a video with a low frame rate. Most tracking methods implicitly assume that a target is close to its position in the previous frame. In our case, this is not true, as displacements may be large. In Fig .1, between (b) and (c), the camera is moved and another person becomes close to the previous target position. In such a case, methods like mean-shift cannot converge toward the target, because they get trapped inside a local maximum. Thus, instantaneous large displacement of the target in the image plane needs to be accounted for explicitely. This is what we want to show in this paper by proposing an adapted tracking framework. In our work, we want to solve the problem of large target displacements in the image plane, a low frame rate, background changes, and tracking with various scale changes. In addition, the tracking algorithm should be able to handle the camera response time, and ultimately zooming. In this paper, we propose an adaptive fuzzy particle filter (AFPF) method adapted to the general object tracking problem with an IP PTZ camera. A fuzzy logic approach is combined with a particle filter that is robust to a non Gaussian distribution of target movements (that is, random motion). Target are modeled and tracked based on sampling around a predicted position obtained by a position predictor, and moving regions detected by optical flow. Sample features are scored based on a fuzzy rule. The fuzzy logic approach is appropriate, in order to avoid hard threshold decision making, and allows us more flexibility in terms of sample score values. Fuzzy membership functions are applied to geometric and appearance features.

In this paper, we apply the AFPF to a human tracking application in an IP PTZ surveillance system. Although applied to track the human upper body, our proposed methodology for working with an IP PTZ camera is general, in the sense that it models the camera system as a servo control loop, and accounts for camera response time and network delays characteristic of such a camera. It can also be extended to other features or targets by adapting the target model.

Results show that our system has a good target detection precision ($> 93.9\%$), low track fragmentation, and a high processing rate, and the target is almost always located within $1/6^{th}$ of the image diameter from the image center. In addition, results show that our method can cope with large displacements from the target in the image plane. In fact, it selects the most likely target among the candidate blobs found everywhere in the image using motion detection.

Our proposed method is:

- capable of supporting large displacements by the object in the image plane;
- online and capable of handling Internet delays and camera response time;
- capable of recovering tracking, in the case of occlusion or a lost target under some conditions.

Section II gives an overview of the state of the art. Section III describes the servo system architecture. Section IV presents our proposed upper body tracking algorithm, and section V presents and discusses validation experiments. Section VI concludes the paper.

II. Related works

The main problem we have to solve is object localization with a large target displacement in the image plane between two consecutive frames and with backgound motion in a low-frame-rate tracking system. Most works to date on tracking with a PTZ camera does not take into account both these difficulties. These works can be divided into three groups: 1) tracking with a PTZ camera and a static camera, in this case the difficulty related to background motion being handled by the static camera, but at the expense of calibration between cameras; 2) tracking with a PTZ camera using background subtraction, in this case the background motion being handled by learning the background for all positions of the camera, but at the expense of a complex setup; and 3) tracking only with a PTZ camera, which is the most difficult case, and in general does not take into account a low frame rate. In fact, most methods account for background motion, but not for low frame rate, and methods that consider a low frame rate do not take into account for background motion. Here, we must consider both. We also need to consider tracking under a low frame rate, as well as the specific case of particle filter trackers.

A. Tracking with a PTZ camera and a static camera

Various combinations of a PTZ camera with multiple PTZ or stationary cameras in a master-slave configuration to explore the field of view have already been studied [1]–[6]. The important issue in such a combination is to find the geometrical relationships between the cameras. To address it, camera calibration is an appropriate solution that is applied in [1], [7], [8]. In these typical works, a system contains distributed stationary and PTZ cameras that are controlled by visual tracking algorithms with a central supervisor unit. Calibration between the cameras reveals the geometrical relationships between them. Moving objects are detected and tracked by the stationary camera, and the PTZ camera is commanded to follow the target from the tracking results of the stationary camera. For example, Krahnstoever *et al.* [5] designed a real-time control system of active cameras for a multiple camera surveillance system. Object are tracked is done by stationary cameras using a shape-based method [9], which detects and compares the human body shape in consecutive frames. The cameras are calibrated using a common site-wide metric coordinate system described in [10] and [11]. The target coordinates obtained are transformed to the appropriate pan and tilt values using geometrical transformations, and the camera moves accordingly. Further, Elder *et al.* [12] suggest a face tracking method for a wide field of view, in which two cameras are used, one a stationary, preattentive, low resolution wide-field camera and the other a mobile, attentive, high resolution narrow-field camera. The face is detected with the low-resolution camera and then the mobile camera tracks the face. Their algorithm for head detection consists of three steps: skin detection, motion detection, and foreground extraction. Funahasahi *et al.* [13], [14] have developed a system for tracking the human head and face parts by means of a hierarchical tracking method using a stationary camera and a PTZ camera. First, the irises are recognized from motion images, as long as the face is large enough to detect them. Then, those irises are used as features for face detection.

In these works, since many cameras share the same field of view, the PTZ camera used to follow the target does not need to predict its position, but simply moves based on the position of the target in other camera's field of view. Thus, it is a different problem from the one we are studying here, where the PTZ camera must track a target without any other information than what it can extract from its own field of view. The advantage of object tracking using only one PTZ camera is that the requirement of the stationary cameras is removed to reduce the cost of the surveillance system, while maintaning a wide coverage of the scene.

B. *Tracking with a PTZ camera and background subtraction*

Adaptive background generation with a geometric transform-based mosaicing method may also be used for person tracking with a single PTZ camera [15]. For each consecutive frame, it finds the best features for the correspondence and then tries to shift the moved image and update the changed background. This approach is based on a high cost background modeling using a calibration scheme, which is not suitable for tracking by Internet-based PTZ cameras. Yao *et al.* [16] propose an algorithm for the scale estimation of 3D objects for zooming purposes with a linear solution based on an affine projection model. For tracking, they apply a dynamic memory method [17]. In their method, pan and tilt angles are obtained from an active background subtraction method. They prepare a recorded background image database in a dynamic memory that contains all the background images at different pan and tilt values. Then, they interpolate the image that they want to process with the image database to obtain the approximate pan and tilt values corresponding to the current image. This method requires that a large number of images be recorded first and then processed. Lim *et al.* [6] propose another alternative with such a setup. To extract the foreground regions (blobs) that contain the movement of the target, they apply an adaptive background subtraction method [18]. The blobs are tracked by finding the correspondence between the blobs of two consecutive frames using an edge matching algorithm. A Kalman filter is used to improve the tracking results and predict the location of the objects. They then map this position to the other camera to pan and tilt.

In these methods, the background motion problem is solved at the expense of complex initialization and background updating computations. If the camera is moved, the learned backgrounds are no longer valid. Thus, this approach lacks flexibility.

C. *Tracking with a PTZ camera only*

Roha *et al.* [19] propose an edge-based object tracking method, and use face tracking as an application of their technique. Accurate tracking is achieved by selecting only boundary edge pixels using optical flow, but this comes at a high computational cost in real-time applications. Venkatesh Babu *et al.* [20] combined two tracking methods: the sum of squared differences (SSD) and the color-based mean-shift (MS) tracker [21]. SSD compares the object appearance frame by frame, by minimizing the intensity sum of squared differences of the objects in two consecutive frames like a block matching process. In testing the methods in [19] and [20], the camera is moved manually and not automatically, and it merely follows a predetermined trajectory. In addition, in their work, the face region is large.

Qu *et al.* [22] have developed a control-based object tracking framework that unifies some kernel-based approaches, such as mean-shift, into a consistent framework. Tracking is modeled as a recursive inverse problem. But kernel-based methods, such as mean-shift, fail to handle large displacement in the image plane, nor, as they mention in their paper, can it deal with severe occlusions. Also, the method has a high computational cost.

Bagdanov *et al.* [23] proposed an active face tracking method using Viola and Jones' face detector [24] to detect face regions with the Shi-Tomasi feature point tracker to track the face. They limit camera movement to only eight cardinal directions and perform a continuous zooming at a constant velocity for a prespecified constant amount of time.

In most of these works, a high frame rate is important for feature consistency (small instantaneous appearance change) and small object displacement between frames (restricted search area). This is why low frame rate methods have been studied explicitly.

D. *Tracking under low frame rate*

Li *et al.* [25] developed a cascade particle filter method with discriminative observers of various life spans for low frame rate videos. Each observer or observation model (such as a two-frame template matching tracker) should be learned from different ranges of samples, with various subsets of features (for example, Viola and Jones' face detector). Their method needs a learning step that is based on model complexity and that increases computation time. The method has limitations in terms of distinguishing between different targets, and has problems with overupdating of the model. Recently, Leichter *et al.* [26] proposed an algorithm for visual tracking under general conditions. The algorithm works by maximizing the PDF of the target's bitmap, which is formulated by the color and location of pixels at each frame. Tracking is based on three assumptions: color constancy, spatial motion continuity and spatial color coherence or similarity of the objects between two consecutive frames. However, severe occlusions are not handled by this algorithm and it is not very fast.

The method we propose is related to those described above, as our aim is to maximize the similarity of the target's appearance in consecutive frames without strong consideration of spatial proximity, but, in addition, it considers a PTZ camera, and thus requires motion prediction and robustness to background changes. Our objective in this work, is to solve the problem of large displacements by targets in the image plane, low frame rate, background changes, and tracking with various scale changes. In addition, the tracking algorithm should be capable of handling camera response time and zooming.

E. Particle filter human tracking

The particle filter has been widely applied in people detection and tracking in many applications, such as indoor and outdoor video surveillance, teleconferencing, multimodal user interface, and lip reading with different camera types. For example, Zhou *et al.* [27] propose an audiovisual detection and tracking system to detect and track the speakers in multimodal user interface application. The visual part of the system is a particle filter-based tracking method and is used for sensor network applications. Their audiovisual tracker summarizes the scene by producing metadata to communicate faster between network nodes in low bit rate communication. The cameras are assumed to be stationary. Pan *et al.* [28] suggest an optimal particle filter method that has several advantages over the classical particle filter, such as smaller tracking distortion, better tracking quality, and less CPU use. Their method has been tested on indoor and outdoor videos using stationary cameras for people tracking.

Vadakkepat *et al.* [29] improved the particle filter to track a randomly moving object using a mobile robot equipped with a pan-tilt camera and 16 sonar sensors. The particle filter uses the sonar information to detect and track the objects, and generates samples everywhere to detect the target. Its motion is random and the background changes according to the camera movement.

Zhai *et al.* [30] presented a novel particle filter method for human tracking. Their particle filter incorporates multiple models (MM) and state partitioning with parallel extended Kalman filters (EKF) to obtain further estimation accuracy. In fact, at the sampling stage, the particle filter uses the information generated by MM and EKF at the observation stage. The algorithm has been tested on videos recorded by stationary cameras and with cluttered backgrounds containing camera ego-motions.

Sommerlade *et al.* [31] presented a scene exploration method combined with zoom control. They minimize the uncertainty of object locations using a Kalman filter tracker, and assumed a simple constant velocity target position change. Similarly, Ahmed *et al.* [32] have proposed an edge-based tracking method that compares the correlation of the target model edge with an image edge to find the target. They predict the next target position using a Kalman filter. The problem with the Kalman filter is the poor estimation of state variables when the ditribution is non Gaussian. The Kalman filter is used under the assumption of a Gaussian distribution of noise and a linear relation for system dynamics. Since the tracking of human movement is non linear and non stationary, the Kalman filter assumption leads to false tracking results.

In [33], object tracking is addressed by first performing moving object detection and then continuous zooming on the detected object without changing pan and tilt values. In the first phase, a set of features of the moving target is tracked using the Shi-Tomasi-Kanade method [34], [35]. In the second phase, the camera starts to zoom in on the moving target. Their method fails in tracking conditions with more than one moving object in a scene or if the target moves out of the field of view of the camera. Similarly, Schreiber [36] propose a histogram-based tracking method, which is, in fact, a generalized template matching Lucas-Kanade algorithm. Zhou *et al.* [37] have introduced an object tracking method using SIFT features and the mean-shift algorithm. SIFT features are used to put in correspondence the regions of interest between frames, and the mean-shift algorithm is used to find the search region similarity. Their tracking method fails if the target has random motions between frames. Also, in their experiments, the camera is assumed to be stationary, and therefore the appearance of the scene does not change too much between frames. This helps the SIFT tracker to better match the corresponding regions of interest.

In an early version of our work [38], we introduced a fuzzy feature-based method for online people tracking. It detects candidate targets in every frame by extracting moving targets using optical flow, an ad-hoc sampling method, and appearance. The target is detected among samples using a fuzzy classifier. Results were shown to be promising. However, the processing rate was low, and objects could be lost or not well localized. In this paper, we have extended and improved this previous work by formalizing the sampling method in the particle filter framework. Target modeling and tracking are now accomplished based on sampling around a predicted position obtained by a position predictor, and moving regions detected by optical flow. This results in better object tracking precision and less target loss. We have also modified the target model (vertical partitioning of samples) to make it more robust and track a broader variety of targets (e.g. cars, buses, animals, etc.) and added more features to our fuzzy classifier (e.g. histogram intersection). We also improved the servo system by considering the image processing delay and by enhancing motion prediction and camera control.

III. System architecture

We consider our PTZ camera as a servo control system. From the system classification point of view, the system is discrete, stable, time variant, causal, non deterministic, and dynamic. These properties help us in modeling the system, which is causal and thus its response at time t depends only on values occurring currently or that occurred in the past. This causality information helps us predict the current situation. Also, the system is discrete, which means that we do not capture images continuously and there is a delay between images modeled by a delay block τ_1. The system is stable, which means that, for a finite input (previous pan-tilt-zoom values), there is a finite output (the next pan-tilt-zoom values). The system is time-variant and varies according to the scenario, processing time, network traffic delay, pan and tilt change delays, and other events. The system is non deterministic, because of the randomness of the sampling process of the particle filter. It is a dynamic system that is constantly changing and its value depends on the current and past input values.

The servo control and tracking system are modeled by a closed-loop control that has a negative feedback, as shown in Fig. 2. The servo system consists of five main blocks: initialization, image capture, object tracking, position prediction, and camera control. Initialization extracts the target at the initial pan and tilt angles (θ_0, φ_0) and zoom (ξ_0), and models it as the input $M(\theta_0, \varphi_0)$ of the Adaptive Fuzzy Particle Filter (AFPF).

AFPF is a fuzzy particle filter tracker adapted to our tracking problem, which performs object tracking and position prediction, as shown in Fig. 2. It is adaptive in the sense that the tracking system can work for both stationary and moving cameras. AFPF is affected by three delays: τ_1 from image capture, τ_2 in the feedback loop from executing camera motion commands, and τ_3 from object tracking. τ_1 depends on network traffic, but is almost constant for particular network traffic at about 0.05s for a 640 $\times$ 480 image. τ_2 is affected by specific camera motion and varies in the range of [0.71s, 2s], depending on the network traffic and on the pan, tilt, and zoom values (that is, the motion amplitude, because of PTZ motor speed constraints). τ_3 is the processing time of the fuzzy particle filter tracker, and varies according to the number of samples, the size of samples, and the number of moving pixels calculated from optical flow.

The input of the system at each time step t consists of the current pan and tilt angles (θ_t, φ_t) and the zoom (ξ_t) of the camera. The output will be the pan and tilt angles and the zoom determined by the fuzzy classifier. The delays imply that the current position of the target cannot be used for centering the camera. To compensate for motion of the target during the three delays, a position predictor is designed inside the AFPF. The position predictor will estimate and predict the next target location according to the previous target motion vectors and previous average system delays. Because of the latency of the network and the camera response, it is not possible to control the camera continuously. Camera commands are queued and performed one after the other, and, if a new command is sent and the queue is not empty, this command will most probably be applied too late and target will be lost. During camera movement, new images are captured and processed to find the target (i.e. images are blurry due to the motion of the camera Fig. 6 (b)). Then, the target position is predicted by prediction block; However, before sending the predicted position to the camera, the queue should be checked. In our servo control system, a new motion/zoom command is issued only if the queue is empty. This way, good control of the camera is maintained, but the target may move during the last motion/zoom command. Thus, the target motion needs to be estimated to keep the target inside the camera's field of view (FOV).

IV. Methodology

The algorithms used for the AFPF tracker and camera control are explained in this section. The AFPF tracker consists of object tracking and position prediction. The target is localized continuously by the object tracker and its position is sent to camera control after system latency compensation from the position predictor. Camera control is used to center the PTZ camera on the target with adjusted pan, tilt, and zoom values. All these steps are explained below, but we first list the assumptions we have made for our application to human upper body tracking:

- participants walk at a normal pace or quickly, but do not run (max: $2.5 m/s$ at $10 m$ from the camera);
- participants do not resemble one another;
- a wide FOV (approximately $48\,^{\circ}$) and scenes that are not crowded (maximum 2-3 people);
- indoor scenes with only small variations in illumination or light sources.

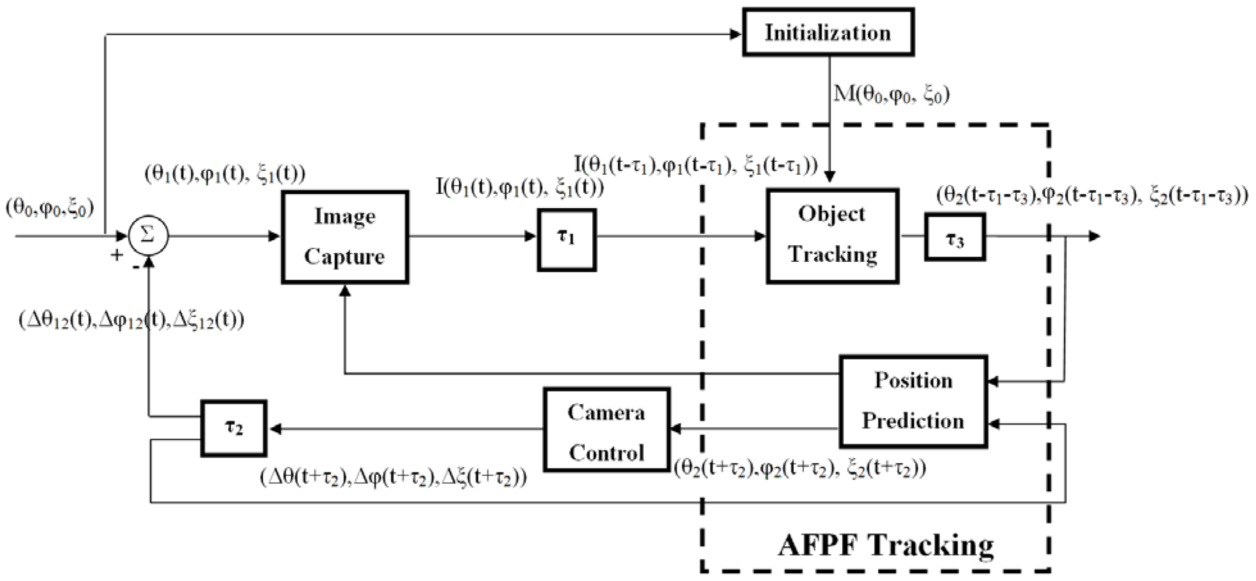


Fig. 2. The system architecture and servo control model. (θ_0,ϕ_0,ξ_0):initial pan, tilt and zoom values,($\Delta\theta_{12}$,$\Delta\phi_{12}$, $\Delta\xi_{12}$) means (θ_1-θ_2,ϕ_1-ϕ_2, ξ_1-ξ_2), and I($\theta_i(t)$,$\phi_i(t)$,$\xi_i(t)$) means image at time t for angles (θ_i,ϕ_i) and zoom $\xi_i(t)$.

A. Adaptive Fuzzy Particle Filter tracker

We developed a fuzzy particle filter tracker that is adapted to our tracking problem. The particle filter is a Bayesian method that recursively estimates the state of the tracking target as a posterior distribution with a finite set of weighted samples [39]. It operates in prediction and update phases. A sample is a prediction of the state of tracking target. In a particle filter, candidate image regions (samples) are compared with the target region model using measurements to select the most likely candidate as the tracked object. Since many features may be used as measure of likelihood, they are combined and normalized using a weighting scheme prior to comparison. In this work, we use fuzzy logic as the weighting scheme to select samples proportionally to their probability of being the target. A fuzzy logic approach allows more flexibility in the weighting because normalization might be done by any function that takes a measure and transforms it to a score between 0 and 1. For example, the camera is always moved to center on the target. Thus, normally, the target location should be around the image center if it motion is well predicted. Thus, a Gaussian membership function is an appropriate choice in this case, because a candidate image region in the center of the image is more likely to be the target. Fuzzy membership functions are applied to geometric and appearance features. Candidate regions are selected based on sampling around the predicted position obtained by the position predictor block and moving regions detected by optical flow.

1) Observation model and initialization: To represent the target, it must be modeled by some features, and these features should be invariant, discriminating, and sufficiently stable during tracking. These constraints are dependent on the application and on tracking scenarios. In our application, the tracking system has a low frame rate, under approximately constant illumination, but with variability in the background because of pan and tilt changes, with

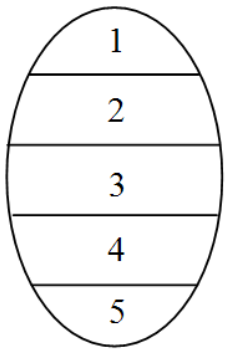


Fig. 3. Vertical partition of ellipse samples.

scaling variations, and with large target displacement variations in the image plane. Although simple, a color-based model satisfies these requirements. Other models could be used with our methodology if more precise details need to be taken into account. However, the model must be robust to instantaneous large scale and viewpoint changes because of the low frame rate. This is consistent with the rationale of the work in [26], which also deals with a low frame rate.

In our problem, the target is represented by an ellipse that circumscribes the head and torso of the human body. The state of the particle filter at each time t is defined as a vector $\vec{P_t}$ of the ellipse's center coordinates $c(t)$, its height $h(t)$, and width $w(t)$, as

$$\vec{P_t} = (c(t), h(t), w(t)). \tag{1}$$

The following geometric and appearance-based features are used as the observation model to represent the target for our particle filter tracker:

- quantized normalized HSV color histogram with 162 bins;
- mean of the H, S and V color components of the HSV color space of all the pixels inside the region;
- center coordinates of each sample or candidate ellipse.

We are also dividing each ellipse horizontally into b smaller regions (e.g. $b = 5$, obtained experimentally according to the approximated ratio of face length to torso length.) to compare the elliptical region more locally (Fig. 3). Indeed, we are interested in having a more accurate semi-local comparison than just a general comparison of the whole sample region. The samples are divided vertically, since we are tracking a human, and so, if the target turns from left to right or from right to left around its main axis, the vertical content of the small regions does not change.

Dividing the region vertically is one of our strategies for handling partial occlusions. We also use the histogram intersection distance for this purpose. For some robustness to illumination changes, we use the HSV colorspace, but are assuming that only small changes occur in the illumination, as stated at the beginning of section IV.

The state vector $\vec{P_t}$ is determined in the initialization step. Currently, initialization is performed manually by selecting the top part of the body (head and torso) of the person to be tracked. This part should always be visible, either when the person is far away from the camera or when he is close to it. The torso constitutes $1/3$ of the total height of the body, as considered in [40]. We fit an ellipse inside the bounding box of the selected region, and model the resulting elliptical region with the previously described features. This is the initial target model M.

Fig. 6 (a) and (e) show the elliptical region (torso and head). We use an elliptical region, because it fits the shape of the head and torso better. To extract the pixels inside the elliptical region from the selected rectangular region, a mathematical rule is applied [41].

2) Sample scoring using a fuzzy classifier: To localize the target, the features of each sample s_i are compared with the initial model M, and a weight or a score ω_i is given to each s_i using a fuzzy rule. Various measures are applied on both whole and divided sample regions. The goal of our fuzzy classifier is to normalize and weight the most discriminative measures to compare the target model features robustly with sample features. We apply different measures on color, because each of them addresses a different level of accuracy. We want to balance the precision of measures in general conditions. We use the following measures to weight the samples:

1) $\varphi_{EC}(s)$, the Euclidean distance between mean HSV of the sample s, ($\bar{H}(s), \bar{S}(s), \bar{V}(s)$), and mean HSV of M, ($\bar{H}_m, \bar{S}_m, \bar{V}_m$) is the first appearance and is defined as:

$$\varphi_{EC}(s) = \sqrt{(\bar{H}(s)-\bar{H}_m)^2 + (\bar{S}(s)-\bar{S}_m)^2 + (\bar{V}(s)-\bar{V}_m)^2}. \tag{2}$$

It is normalized to between 0 and 1 by a linear membership function. Since each element of the HSV histogram (H, S, and V) varies between 0 and 255, the maximum value of $\varphi_{EC}(s)$ is $255\sqrt{3}$. Therefore the normalization function of $\bar{\varphi}_{EC}(s)$ is given by:

$$\bar{\varphi}_{EC}(s) = 1 - \frac{\varphi_{EC}(s)}{255\sqrt{3}}. \tag{3}$$

This distance is also applied on the divided regions of each sample, the corresponding membership functions of which are $\bar{\varphi}_{EC1}(s)$, $\bar{\varphi}_{EC2}(s)$, $\bar{\varphi}_{EC3}(s)$, $\bar{\varphi}_{EC4}(s)$, and $\bar{\varphi}_{EC5}(s)$. The HSV color space is robust to illumination changes. The distance is a color-based comparison of the average color content of each sample with the target model to compare the similarity of the sample's appearance with that of the target.

2) $\varphi_{EP}(s)$, the Euclidean distance between the sample center coordinates ($x(s), y(s)$) and the video frame center coordinates (x_{im}, y_{im}) is a geometric measure, and is defined as:

$$\varphi_{EP} = \sqrt{(x(s)-x_{im})^2 + (y(s)-y_{im})^2}. \tag{4}$$

Normally, the person should be near the video frame center, given a good position prediction. Thus, a Gaussian membership function $\bar{\varphi}_{EP}(s)$ is used to normalize it, as:

$$\bar{\varphi}_{EP}(s) = exp(-\frac{{\varphi_{EP}}^2(s)}{2\sigma^2}). \quad (5)$$

σ^2 is equal to a quarter of the image area around the video frame center. This distance is an appropriate measure for evaluating the sample location according to our goal (i.e the camera field of view center should always be near the target).

3) $\varphi_{EH}(s)$, the Euclidean distance between quantized normalized HSV color histogram of sample s, $His(s)$, and the histogram of the initial target model M, His_m [42], is defined as:

$$\varphi_{EH}(s) = \sqrt{\sum_n (His(s)[n] - His_m[n])^2}, \quad (6)$$

where n is the histogram bin number. Since the HSV histogram is normalized, the maximum value of $\varphi_{\ EH}(s)$ is $\sqrt{2}$ and the minimum value is 0. Therefore, the membership function $\bar{\varphi}_{\ EH}(s)$ is:

$$\varphi_{EH}(s) = 1 - \frac{\varphi_{EH}(s)}{\sqrt{2}}. \quad (7)$$

As discussed in the above, color is one of the appearance features. The color histogram of a region represents the color distribution of that region, and is a more precise color appearance similarity measure. This membership function is also defined for the divided regions, as $\bar{\varphi}_{\ EH1}(s)$, $\bar{\varphi}_{EH2}(s)$, $\bar{\varphi}_{EH3}(s)$, $\bar{\varphi}_{EH4}(s)$, and $\bar{\varphi}_{EH5}(s)$.

4) $\varphi_H(s)$, the 2-D correlation coefficient between $H(s)$ and H_m, is given by:

$$\varphi_H(s) = \frac{\sum_n((H(s)[n]-\bar{H})(H_m[n]-\bar{H}_m))}{\sqrt{\sum_n (H(s)[n]-\bar{H})^2}\sqrt{\sum_n (H_m[n]-\bar{H}_m)^2}}, \quad (8)$$

where $\bar{H}$ and $\bar{H}_m$ denote the average of $H(s)$ and H_m. The 2-D correlation coefficient varies between -1 and 1, therefore its membership function $\bar{\varphi}_{\ H}(s)$ is defined as:

$$\bar{\varphi}_H(s) = \frac{1 + \varphi_H(s)}{2}. \quad (9)$$

$\varphi_H(s)$ is our most precise similarity measure, and is used to find the correlation between two color distributions. This membership function is also defined for the small regions as $\bar{\varphi}_{\ H1}(s)$, $\bar{\varphi}_{H2}(s)$, $\bar{\varphi}_{H3}(s)$, $\bar{\varphi}_{H4}(s)$, and $\bar{\varphi}_{H5}(s)$

5) $\varphi_{HI}(s)$, the intersection of histograms $H(s)$ and H_m, is the last appearance similarity measure. It is defined as:

$$\bar{\varphi}_{HI}(s) = \sum_n min(H(s)[n], H_m[n]). \quad (10)$$

Since $\varphi_{HI}(s)$ is between 0 and 1, $\bar{\varphi}_{HI}(s)$ is equal to $\varphi_{HI}(s)$. This membership function is also defined for the divided regions as $\bar{\varphi}_{HI1}(s)$, $\bar{\varphi}_{HI2}(s)$, $\bar{\varphi}_{HI3}(s)$, $\bar{\varphi}_{HI4}(s)$, and $\bar{\varphi}_{HI5}(s)$.

The weight (score) of the sample i at time t, ω_i^t, is obtained by adding all the membership function values of the whole sample and five divided regions, as follows:

$$\omega_i^t = \tag{11}$$
$$\bar{\varphi}_{EC}(s_i^t) + \bar{\varphi}_{EP}(s_i^t) + \bar{\varphi}_{EH}(s_i^t) + \bar{\varphi}_{H}(s_i^t) + \bar{\varphi}_{HI}(s_i^t)$$
$$+ \sum_5^{j=1} \bar{\varphi}_{ECj}(s_i^t) + \bar{\varphi}_{EHj}(s_i^t) + \bar{\varphi}_{Hj}(s_i^t) + \bar{\varphi}_{HIj}(s_i^t).$$

Target s_f is the sample at each time t that has the maximum weight, and is selected by:

$$s_f = argmax_{s_i \in S} \left\{ \omega_i^t \right\}. \tag{12}$$

3) Re-sampling and updating: Among all N samples in each frame, N_{s_i} samples with high probabilities (weights) are selected (i.e. here $N_{s_i} = 1$). Thus, the current sample set S_t is determined by the replacement of N_{s_i} samples with higher probability from the N samples of the previous sample set S_{t-1} at time $t-1$,

$$S_{t-1} = \left\{ c_i^{t-1}(h_i^{t-1}, w_i^{t-1}), \omega_i^{t-1} \right\}_{i=1}^{N} \tag{13}$$

where c_i^{t-1} are the i^{th} sample coordinates with height and width h_i^{t-1} and w_i^{t-1} respectively at time $t-1$, and ω_i^{t-1} is the related weight (score) to the i^{th} sample. Sample set S_{t-1} is an approximation of the posterior distribution of the target state at time $t-1$.

The particle filter state in two consecutive frames does not change significantly if the camera does not move. Here, it is a translation of sample coordinates around its previous position and scaling of the previous sample size. No rotation is applied in our application, since people are assumed to be walking upright.

At each time t, samples are reproduced in the state space by a dynamic first-order auto-regressive model given by:

$$\vec{P_t} = \vec{P_{t-1}} + \omega_t. \tag{14}$$

$\vec{P_t}$ and $\vec{P_{t-1}}$ are the particle filter states at time t and $t-1$ respectively. ω_t is a multivariate Gaussian random variable, and it correlates to a random translation of the sample corner coordinates and scaling of the previous sample size.

Using only the previous sample set is not appropriate in our application, because the camera is moving and the position of the previous state has changed. Therefore, from the previous sample set, we only resample around the sample with the highest probability s_f, if the camera does not move. But, if the camera moves, since it is assumed that the target is always at the center of the image, we only sample around the image's center position with the last

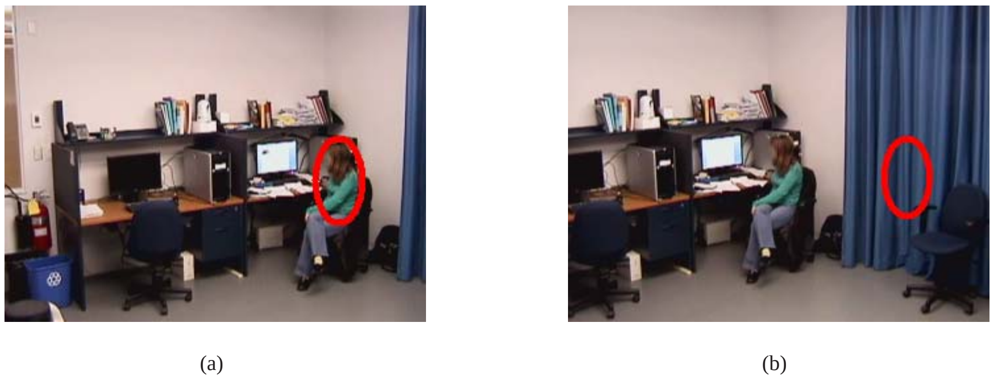


Fig. 4. Target sample in two consecutive frames (a) before moving the camera, and (b) after moving the camera.

target sample size during the camera movement to generate N samples. During that camera movement, the image center is considered a prediction of the target position, since ideally the target should always be at the image center. Fig. 4 shows two consecutive frames during camera movement. If we continue to resample around a previous target position instead of the image center, we will do the sampling incorrectly.

To our adapted particle filter tracker, we have added coordinates of the moving regions extracted by optical flow to our sample set at time t. So we sample the image with ellipses around two types of regions of interest, which are:

1) the area around the previous target position coordinates or around the image center,
2) the moving areas extracted by optical flow.

If the camera starts to move, the sampling process is performed around the image center and moving areas extracted by optical flow. The second type of sample is detected by estimating the motion of the target from two consecutive images I_t and I_{t+1}, using pyramidal Lucas Kanade optical flow [34]. In [34], strong corners in the image that have large eigenvalues are detected for comparison. To solve the pixel correspondence problem for a given pixel in I_t, we look for nearby pixels of the same color in I_{t+1}. The basic concept in optical flow is to find the motion vector d for two consecutive images I_t and I_{t+1} by minimizing the following residual function ϵ:

$$\begin{aligned}\epsilon(d) &= \epsilon(d_x, d_y) \\ &= \sum_{x=u_x+w_x}^{x=u_x-w_x} \sum_{u_y+w_y}^{y=u_y-w_y} (I_t(x,y) - I_{t+1}(x+d_x, y+d_y))\end{aligned} \tag{15}$$

where $(2w_x+1) \times (2w_y+1)$ is the integration window size for evaluating. Usually, w_x and w_y are equal to 2, 3,...,7 pixels (here, we selected 3, based on experiments). For the pyramidal representation of the images, this residual function will be minimized at each level. We use 4 pyramid levels with 10 iterations. The threshold for stopping iterating for minimization of the residual function is 0.3. Applying optical flow extracts motion-based pixels with their related motion vector results from camera movement or object movement.

As found experimentally, the detected motion vector results are noisy. In addition, the camera motion vectors have an effect on the object motion vectors. To remove this effect, camera motion vectors must be extracted. To calculate the camera motion vectors, a radial histogram of motion vectors is calculated. In a radial histogram, each bin is based on the quantized length (r) and angle (θ) of the motion vector. Our radial histogram, $h(r, \theta)$ has 36180 bins. r has 200 values and is varied based on image size between 1 and the image diagonal (e.g for an image size 640 $\times$ 480: $r = 4$ pixels, and θ has 180 values and is varied between 0° and 360° ($\theta = 2^\circ$)). These values are obtained experimentally. The r and θ of the bin that has the maximum number of vectors are designated as the camera motion vector length and angle respectively. The detected motion vectors with this length and angle are removed, and then the camera motion vector is subtracted from the rest of the motion vectors using minimum bin values.

Motion vectors are then grouped according to their distances from each other and their length and directions. Motion-based samples are extracted around object motion vector groups.

4) Position prediction: As discussed, the three delays in the control loop cause the system to fail to center correctly on the target. To compensate for target motion during the delays, a position predictor based on the D last motion vectors of the target (i.e. when the camera is not moving) has been designed. This motion predictor will consider the angle between D ($D = 3$) consecutive motion vectors. If the angle difference is smaller than ϑ° (i.e. $\vartheta^\circ = 25^\circ$, obtained experimentally), it is assumed that the target is moving in the same direction and its average speed can be estimated. The average speed of the target $\overline{\nu}$ for the D last target motion vectors is thus equal to:

$$\overline{\nu} = \frac{\sum_{i=1}^{D} \Delta x_i}{\sum_{i=1}^{D} (\tau_1^i + \tau_3^i)}, \tag{16}$$

where Δx_i, is the i^{th} target displacement vector (i.e. target motion vector). τ_1^i is the i^{th} delay τ_1 and τ_3^i is the i^{th} delay τ_3 associated with each motion vectors. Thus, the system will put the camera center at the predicted position, which is:

$$c_p = c_E + \overline{\tau}_2 \times \overline{\nu}, \tag{17}$$

where c_p is the predicted target coordinate and c_E is the extracted target coordinate from object tracking. $\overline{\tau}_2$ is the average delay time τ_2 obtained in proportion to displacement Δx_{C_E} required to move to the coordinates c_E. Each camera movement creates a delay of τ_2^i where i is the i^{th} movement of the camera with a displacement of $\Delta x_{C_E}^i$. $\overline{\tau}_2$ is calculated based on the average time required for all previous camera displacements (e.g. all $\Delta x_{C_E}^i$).

$$\overline{\tau}_2 = \Delta x_{C_E} \times \frac{\sum_i \tau_2^i}{\sum_i \Delta x_{C_E}^i} \tag{18}$$

$\overline{\tau}_2$ is the time required to move the camera by Δx_{C_E} multiplied by the mean time required per unit displacement. Thus, the camera is controlled and moved adaptively with the speed of the network, camera response time, and displacement vector of the target. Our tracking algorithm can be summarized in the following steps:

1) Initialization
   - Manually select a rectangular region around the upper body and then automatically fit an ellipse over the target head and torso.
2) Prediction
   - If the camera does not move, resample around the sample with highest score from samples at time t-1, s_f, based on the probabilities determined by Eq. 12.
   - If the camera is moving, resample around the image center with the previous target size.
   - For two consecutive images, apply optical flow and extract groups of object motion vectors from camera motion vectors. Resample around the group coordinates.
3) Update
   - Update new sample weights from observation measurements using Eq. 11.
4) Fuzzy feature scoring
   - Select the best sample using Eq. 12.
5) Position predication
   - Predict the position of the target based on the D last motion vectors of the target.
   - Send the predicted position c_p to the camera.
   - Go to step 2.

B. Camera control

Fig. 5 shows the perspective transform from the 3D world coordinates to the 2D image plane coordinates. To determine which image coordinate $c_P = (C_x, C_y)$ corresponds to a world point P the following relations are used:

$$C_x = P_x \times \frac{C_z}{P_z} \tag{19}$$

$$C_y = P_y \times \frac{C_z}{P_z}. \tag{20}$$

In our case, the camera is controlled based on the target motion on the 2D image plane. That is, we assume that the object is moving on a plane parallel to the image plane ($\frac{C_z}{P_z}$ is assumed constant). Because of the low frame rate, this is not exact, but it is an acceptable approximation for a walking human.

1) Moving criteria: To follow the target, the PTZ motors are commanded based on c_p. As discussed earlier, in our servo control loop, c_p is computed and the camera is moved to keep track of the target while taking into account variable delays. We need to be careful about when to move the camera to avoid losing the object from the field of view. Thus, we use the following criteria: 1) a moving command is sent to the camera if the camera is not moving for the reasons explained in section III, and 2) a moving command is sent if the score of target sample s_f (Eq. 21), that is obtained by AFPF, is higher than $\eta\%$ of the maximum score s_{max}. That maximum score is the measured score assigned to the target in the first frame after initialization. η is obtained experimentally (e.g. η = 85). The moving command will be sent if:

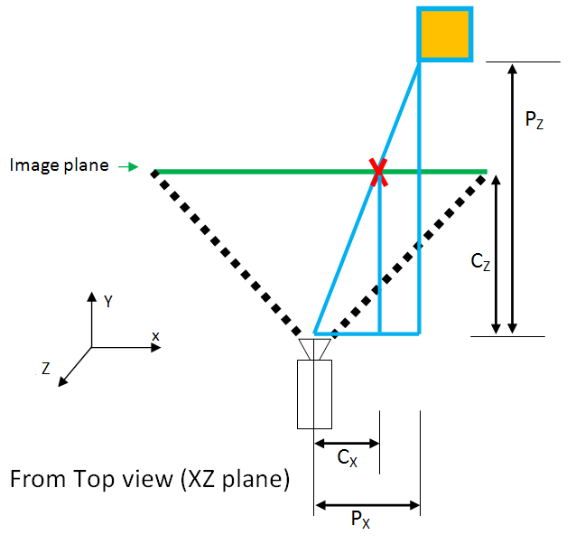


Fig. 5. Perspective transform geometry.

$$s_f \geq \eta \times s_{max} \tag{21}$$

The goal of this criterion is to move the camera only if we have a good level of confidence about the target localization.

2) Camera command: To control the camera movement, we calculate the angles on the workstation and send computed values to the camera using its RelativePanTiltZoom function. In [38], we showed that this function has better performance than the AreaZoom camera control function. It reduces the camera response time, and thus we obtain a faster system rate and the target is more often in the center of the image. Thus, we used the RelativePanTiltZoom function to control and move the camera. The pan and tilt values are calculated on the workstation according to the field of view of the lens and the range of pan and tilt angles for each camera (i.e. they are dependent on the camera model). The angles are sent to the camera by sending an HTTP POST request using the camera's CGI scripts [43] and the *RelativePanTiltZoom* function, for which, it is possible to specify the speed of motion. This speed is set to the maximum value.

V. Experiments and analysis

We first present our experimental methodology, followed by our results, and, finally, a discussion.

A. Data acquisition and ground-truth

Our method has been implemented in C++ using OpenCV and a custom library to control the IP PTZ camera. We used two Sony IP PTZ cameras (SNC-RZ50N and SNC-RZ25N) for our tests. Table I shows the capabilities of the two cameras.

TABLE I

CAMERA SPECIFICATIONS.

Camera Model	$max(R)$	$max(fr)$	P	$max(PS)$	T	$max(TS)$	OZ
SNC-RZ50N	640×480	30 fps	$-170°$ to $170°$	$300°/s$	$-90°$ to $25°$	$90°/s$	26x
SNC-RZ25N	640×480	30 fps	$-170°$ to $170°$	$100°/s$	$-90°$ to $30°$	$100°/s$	18x

$max(R)$: maximum resolution, $max(fr)$: maximum frame rate, P: pan range, $max(PS)$: maximum pan speed, T: tilt range, $max(TS)$: maximum tilt speed, OZ:optical zoom.

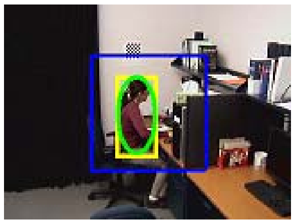
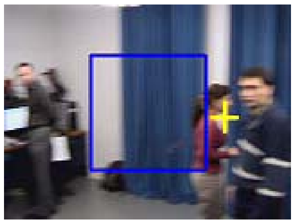
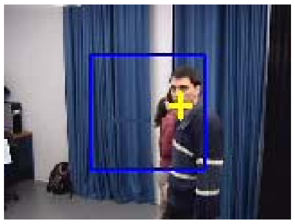
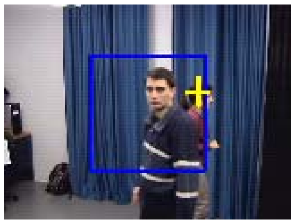
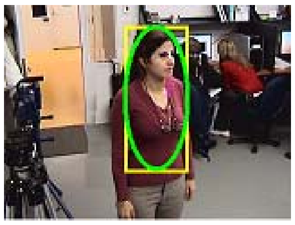
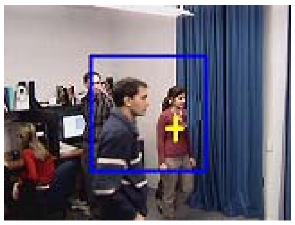
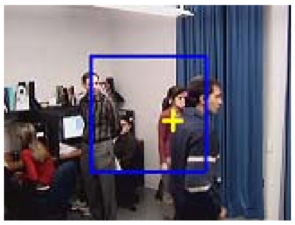
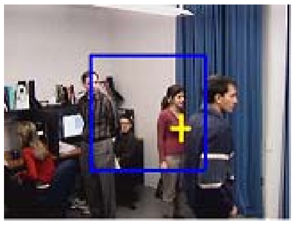
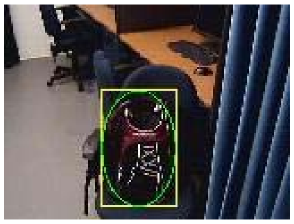
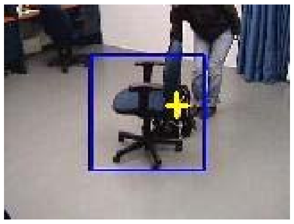
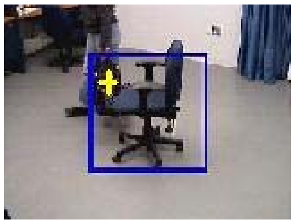
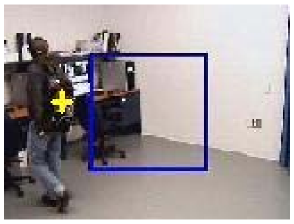
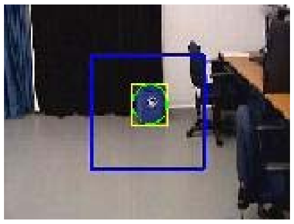
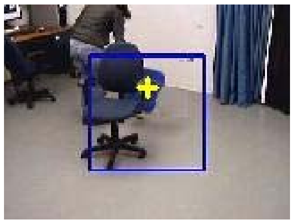
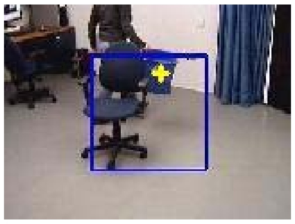
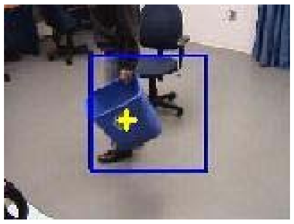

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

(m)

(n)

(o)

(p)

Fig. 6. Examples of tracking frames for E_5 (a) to (d), E_7 (e) to (h), E_{16} (i) to (l), and E_{19} (m) to (p). E_5 (a) initial model selection, (b) before occlusion, (c) during short-term occlusion, (d) after occlusion; E_7 (e) initial model selection, (f) scale variation and before occlusion, (g) during short-term occlusion, (h) after occlusion; E_{16} (i) initial model selection, (j) during short-term occlusion, (k) after short term occlusion, (l) scale variation; E_{19} (m) initial model selection, (n) during short-term occlusion, (o) after short-term occlusion, (p) scale variation. The blue rectangle has a size equal to $1/3^{th}$ of the image diagonal.

For validation, we tested the complete system in online experiments. No dataset is available for testing the complete tracking system, because of its dynamic nature. The tracking algorithm has been tested over events such as entering or leaving the FOV of the camera and occlusion with other people in the scene. We recorded all the experiments to extract their ground-truths manually for performance evaluation. Only usable frames that are captured and processed by the camera are recorded, as explained in section III and section IV. Thus, the recorded video frame rate is not 30 fps, but corresponds to the frame processing rate by the servo control loop. The general scenario of the experiments is the following. An actor is selected (frontal view) for initial modeling. She starts to walk around in a room. From two to four actors can walk at the same time in different directions in the room, crossing paths with the target or occluding her. The target pauses occasionally while walking to verify the performance for the stationary target. The target actor also moves parallel to, toward, or away from the camera. Fig. 6 shows the initial model selection and some frames obtained during tracking.

We performed fifteen experiments with the two IP cameras. The experiments are described in Table II. They are classified into three, based on the camera model, initial model position from the camera, and image resolution. The experiments were carried out in a 6 $\times$ 4 meter room. The distances of the initial model position from the camera are listed in Table II. We compare the effect of various parameters, such as image size, and camera model, in the tracking system performance results.

The AFPF was tested on tracking a recycling bin and a backpack to evaluate the tracking system performance on objects different from a human. The general scenario is the following: First the object is selected, and then a person picks up the object, and walks around carrying it. Next, the object is left somewhere and is occluded by other moving objects or a walking person. Finally, the object is carried parallel to, toward, and away from the camera. Fig. 6 (i) and (m) shows the object initial model selection. For this, we carried out four experiments with SNC-RZ50 and 320 $\times$ 240 image resolution.

B. *Evaluation metrics*

To evaluate our method, five metrics are used:

1) Precision (P) to calculate the target localization accuracy. It is defined as

$$P = \frac{TP}{TP + FP}, \tag{22}$$

where TP and FP are true positive and false positive respectively. TP is the number of frames in which the target is correctly localized by the camera. FP is the number of frames where the target is not localized correctly by the camera.

2) Normalized Euclidean distance (d_{gc}) to evaluate the dynamic performance of the tracking system. It is defined as:

$$d_{gc} = \frac{\sqrt{(x_c - x_g)^2 + (y_c - y_g)^2}}{a}, \tag{23}$$

where (x_g,y_g) is the ground-truth target coordinate and (x_c,y_c) is the center of the image. This is the spatial latency of the tracking system, as, ideally, the target should be at the image center. a is the radius of the

TABLE II

EXPERIMENT TEST CASES.

Classes	Experiments	Camera Model	Image Size	IMP
Class 1	E_1	SNC-RZ50N	640×480	Near
	E_2	SNC-RZ50N	640×480	Near
	E_3	SNC-RZ50N	640×480	Far
	E_4	SNC-RZ50N	640×480	Far
	E_5	SNC-RZ50N	640×480	Middle
Class 2	E_6	SNC-RZ50N	320×240	Near
	E_7	SNC-RZ50N	320×240	Near
	E_8	SNC-RZ50N	320×240	Far
	E_9	SNC-RZ50N	320×240	Far
	E_{10}	SNC-RZ50N	320×240	Middle
Class 3	E_{11}	SNC-RZ25N	320×240	Near
	E_{12}	SNC-RZ25N	320×240	Near
	E_{13}	SNC-RZ25N	320×240	Far
	E_{14}	SNC-RZ25N	320×240	Far
	E_{15}	SNC-RZ25N	320×240	Middle

IMP: Initial model position from the camera, Far: 5 meters, Near: 1 meter, Middle: 3 meters.

circle that circumscribes the image (and corresponds to half of the image diagonal).

3) Normalized Euclidean distance (d_{gp}), which shows the error of tracking algorithm. It is the target position error, and is defined as:

$$d_{gp} = \frac{\sqrt{(x_p - x_g)^2 + (y_p - y_g)^2}}{2a}, \tag{24}$$

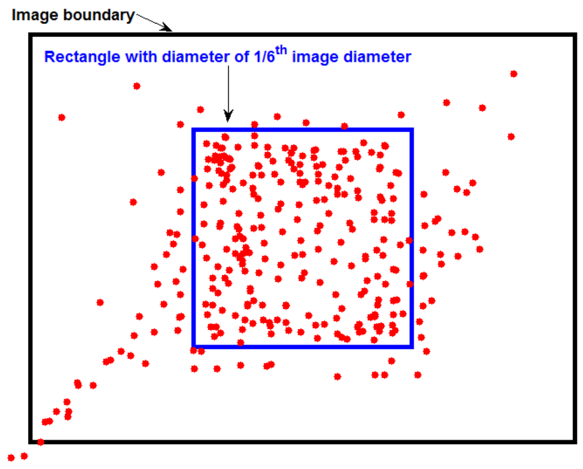
where (x_p,y_p) is the tracked object coordinate. Ideally, d_{gp} should be zero.

4) Track fragmentation (TF) indicates the lack of continuity of the tracking system for a single target track [44].

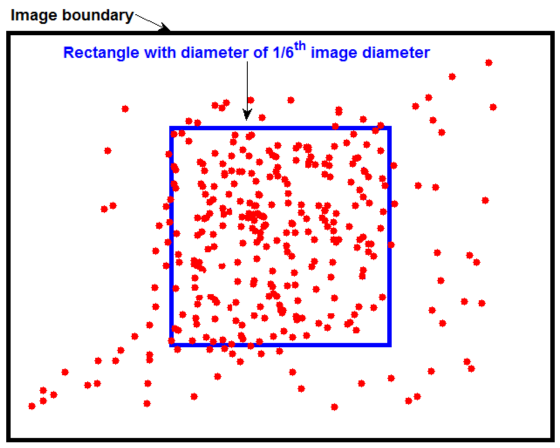
$$TF = \frac{T_{OUT}}{NF}, \tag{25}$$

T_{OUT} is the number of frames where the target is out of the FOV and NF is the total number of frames.

To evaluate the tracking performance, we have compared our method with the classical particle filter (PF) method [45], the CamShift (CSH) tracking method implemented in OpenCV [46] and the Kanade-Lucas-Tomasi (KLT) feature tracking method [34]. To remove the camera control effect from the tracking performance, KLT, CSH and PF are applied separately on the videos recorded during the experiments using the AFPF tracking method. All the experiments are performed first by applying the AFPF, and the sequences obtained are then recorded to be tested by the KLT, CSH and PF tracking methods. For PF, the target is modeled and initialized in the same way as AFPF. We use a sample generator that generates 250 samples by random translations in x and y around the image center in a circular region with a radius corresponding to the image height. The number of PF samples is obtained from the average frame processing rate of our method. We have determined that 250 is the maximum number of samples



(a) E_3



(b) E_{10}

Fig. 7. Target position in the image plane at various frames for (a) E_3, (b) E_{10}

that should be used to arrive at such a frame rate. The size of these samples (width and height of the ellipses) is changed randomly by 5% based on the previous target size.

For KLT, in the first frame, the target is introduced to the tracker by cropping the target silhouette image (Fig.8 (a)). The tracker is then applied to this target area. We kept the default parameters available in version 1.3.4 of the source code. Thus, we kept extraction to a maximum of 40 features in each frame. We also compare the AFPF method results with the results of class 2 of the previous version in [38], since the best results belong to class 2.

For the CamShift tracker, in the first frame, the target is introduced to the tracker by selecting an elliptical region of interest over the face and torso of a person. The tracker can be adjusted by three parameters: V_{min}, V_{max} and S_{min} which are the minimum and maximum value color components and the minimum saturation color component of the HSV color space. V_{min} and S_{min} are used to remove neutral colors (almost gray and almost black) from the histogram computations, and V_{max} to remove pixels that are too bright. We found that the best results of this tracker are obtained by using the default values for these parameters ($V_{min} = 10$, $V_{max} = 256$ and $S_{min} = 30$).

C. Results

Fig. 7 shows the target position in the image plane at different frames for E_3 and E_{10} of Table II. d_{gc} is the normalized distance of each point from the image center. This distance allows us to verify whether or not the tracker has lost the target. If d_{gc} is more than 1, the target has been lost (it is outside of the FOV). For distances smaller than 0.6, the object is inside the FOV. For the range of $(0.6 < d_{gc} < 1)$, it depends on whether or not the centroid coordinates of the target are in the range [0, height-1] and [0, width-1]. For E_{10}, the target is always in the FOV. In E_3, the target was lost twice at frames 69 and 73.

Table III, Table IV, and Table V show the results of the five metrics and the system frame rate for all experiments. For d_{gc} and d_{gp}, we show the mean and variance of all experiments. For classes with a larger image size, the method lost the target several times, but eventually recovered it. For experiments using a smaller image resolution, there are fewer target losses (e.g. smaller TF), less distance error ($\mu_{d_{gc}}$ and $\mu_{d_{gp}}$) and the precision results are better (P is larger). Because of d_{EP} and camera control, the error on $\mu_{d_{gc}}$ has an effect on $\mu_{d_{gp}}$ and vice versa. That is, if $\mu_{d_{gc}}$ is large, the target will have a smaller value for d_{EP}, and in turn the object might then not be localized correctly, increasing $\mu_{d_{gp}}$. Indeed, the algorithm tries to find similar object to the target that is nearest to the image center. Furthermore, if the target moves very quickly, d_{gc} will be increased to minimize d_{gp}. The best results are obtained for class 2, for which the system frame rate is faster. TF for classes 2 and 3 are the smallest. A faster system frame rate improves the results of TF, $\mu_{d_{gc}}$, $\mu_{d_{gp}}$, and P.

By comparing the results of class 2 with those of class 1, the effect of image resolution size for both cameras is evaluated. As shown in Table III, Table IV, and in Table V, with a smaller resolution, a faster frame rate, better tracking performance are obtained. By comparing the results of classes 1, 2, and 3 with those of class 2 in [38], the effect of the previous version of our new method in [38] is studied. Higher values of P with less TF, smaller $\mu_{d_{gc}}$ and smaller $\mu_{d_{gp}}$ values are obtained. In addition, we have a processing frame rate that is twice as fast. Ideally, when the image size drops from 640 $\times$ 480 to 320 $\times$ 240, the frame rate should be four times faster; but here, the system frame rate is also affected by the speed of the camera step motors. Indeed, the mechanical part of the system imposes an upper maximum speed limit for the camera movement, and thus on the system frame rate. This fact is modeled by delay τ_2, which is shown in Fig. 2. By comparing the results for the two cameras, the results of SNC-RZ50 are better than those of SNC-RZ25, because of camera characteristics such as maximum pan and tilt speed, maximum FOV, etc.

Table III, Table IV, and Table V show the five calculated metric values for the PF, CSH, and KLT feature tracker methods in comparison with our method. PF and CSH outperform the KLT tracker, and AFPF outperforms all of them, as shown in Table III, Table IV, and Table V. These tables show the average tracking results of the AFPF tracker for each class compared with the PF, CSH and KLT trackers results. With AFPF, larger P, smaller TF, smaller $\mu_{d_{gc}}$, and smaller $\mu_{d_{gp}}$ are obtained.

Table VI shows the five calculated metric values for the AFPF method with a 320 $\times$ 240 image resolution for tracking of the recycling bin and the backpack objects. By comparing human tracking results (Table III, Table IV, and Table V) with the tracking results of other objects (Table VI), results are obtained that are almost similar (P, TF, $\mu_{d_{gc}}$, and $\mu_{d_{gp}}$ are in the same ranges).

The last two columns in Table III, Table IV, Table V and Table VI are the minimum and maximum length of the target motion vector in number of pixels. These values vary according to the image resolution size, frame rate and target movement.

D. Discussion

Compared to the early version of our work [38], we obtain a higher target tracking precision P with less tracking fragmentation. The localization of the target is improved (smaller $\mu_{d_{gc}}$ and $\mu_{d_{gp}}$ values). This means that the location of the target is near to the ground-truth and the camera could center on the target with a good precision. This result

TABLE III

EXPERIMENTAL RESULTS WITH 640×480 IMAGE RESOLUTION.

E	P (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	TM	$FR(fps)$	NF	$min(\Delta x)$	$max(\Delta x)$
E_1	93	0.2	0.2154	0.0436	0.0521	0.0015	$AFPF$	5.57	511	15	270
	71	8	0.3351	0.1164	0.1022	0.0642	PF				
	47	17	0.6246	0.3151	0.1820	0.1514	KLT				
	56	13	0.5019	0.2487	0.1462	0.1121	CSH				
E_2	96	0.5	0.2521	0.0352	0.0427	0.0021	$AFPF$	6.68	489	11	251
	74	6	0.3618	0.1253	0.1172	0.0527	PF				
	48	13	0.6314	0.3221	0.1931	0.2326	KLT				
	54	11	0.5129	0.2087	0.1643	0.1434	CSH				
E_3	98	0.7	0.1761	0.0271	0.0533	0.0032	$AFPF$	7.31	494	13	289
	76	5	0.3221	0.1096	0.1063	0.0612	PF				
	54	11	0.6117	0.2746	0.1712	0.2238	KLT				
	61	9	0.5081	0.1971	0.1445	0.1578	CSH				
E_4	92	0.1	0.1517	0.0394	0.0426	0.0027	$AFPF$	6.84	506	5	207
	73	7	0.3046	0.1142	0.1158	0.0547	PF				
	51	10	0.6219	0.2231	0.1610	0.2119	KLT				
	66	8	0.5791	0.1762	0.1482	0.1671	CSH				
E_5	91	0.4	0.1729	0.0613	0.0512	0.0023	$AFPF$	6.39	517	7	238
	72	9	0.3003	0.1422	0.1236	0.0435	PF				
	53	12	0.6012	0.2305	0.1522	0.1910	KLT				
	68	10	0.5216	0.1814	0.1232	0.1678	CSH				
Class 1	**93.95**	**0.37**	**0.1936**	**0.0413**	**0.0484**	**0.0024**	**AFPF**	**6.49**	**2517**	**5**	**289**
	73.17	**7.02**	**0.3248**	**0.1215**	**0.1130**	**0.0553**	**PF**				
	50.60	**12.61**	**0.6182**	**0.2731**	**0.1719**	**0.2021**	**KLT**				
	61.06	**10.2**	**0.5247**	**0.2024**	**0.1453**	**0.1496**	**CSH**				

E: experiments, P:precision, $\mu_{d_{gc}}$: mean of d_{gc}, $\mu_{d_{gp}}$: mean of d_{gp}, $\sigma^2_{d_{gc}}$: variance of d_{gc}, $\sigma^2_{d_{gp}}$: variance of d_{gp}, FR: system frame rate, NF: number of frames, $min(\Delta x)$: minimum motion vector length, $max(\Delta x)$: maximum motion vector length, $AFPF$: Adaptive Fuzzy Particle Filter Tracker, KLT: Kanade-Lucas-Tomasi Feature Tracker [47], PF: Particle Filter, CSH: CamShift [46], TM: Tracking Method.

is explained by the use of a particle filter that distributes samples well at each frame in different positions and various scale sizes around the detected motion regions or the previous target position. That gives the tracker and the fuzzy classifier more candidate regions in which to find the best match. Localization accuracy is thus improved. This also results in the target being almost always located at a constant distance within $1/6^{th}$ of the image diagonal from the image center (e.g. the blue rectangle in Fig.7). When localization fails, it is because of the similarity or closeness of the color histogram of the target to other samples. The image resolution and camera model affect the system frame rate, and thus the tracking error.

Comparing the AFPF and PF methods with the CSH and KLT feature trackers, we find that both AFPF and PF have a higher target tracking precision P with less tracking fragmentation, and the localization of the target is better (smaller $\mu_{d_{gc}}$ and $\mu_{d_{gp}}$ values). This means that the location of the target is closer to the ground-truth than the KLT feature tracker results. This is explained by particle filter characteristics that distribute samples well at

TABLE IV

EXPERIMENTAL RESULTS WITH 320×240 IMAGE RESOLUTION.

E	P (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	TM	$FR(fps)$	NF	$min(\Delta x)$	$max(\Delta x)$
E_6	96	0.09	0.1124	0.0271	0.0310	0.0017	$AFPF$	11.63	897	6	180
	81	4	0.1531	0.0721	0.0536	0.0452	PF				
	63	8	0.4121	0.1132	0.1326	0.1001	KLT				
	72	6	0.3209	0.0945	0.1124	0.0782	CSH				
E_7	100	0	0.1230	0.0227	0.0272	0.0018	$AFPF$	12.72	908	3	105
	84	1	0.1627	0.0534	0.0572	0.0367	PF				
	66	9	0.4036	0.1015	0.1406	0.1121	KLT				
	74	7	0.3562	0.0851	0.1261	0.0875	CSH				
E_8	98	0.06	0.0915	0.0231	0.0246	0.0019	$AFPF$	10.29	916	0	169
	86	3	0.1572	0.0682	0.0639	0.0412	PF				
	65	6	0.3941	0.1204	0.1211	0.0992	KLT				
	71	5	0.3087	0.0912	0.1005	0.0801	CSH				
E_9	95	0.1	0.1009	0.0261	0.0263	0.0022	$AFPF$	11.35	903	7	190
	83	5	0.1721	0.0640	0.0644	0.0427	PF				
	69	11	0.3849	0.0984	0.1508	0.0973	KLT				
	75	9	0.2997	0.0762	0.1226	0.0890	CSH				
E_{10}	97	0.07	0.1386	0.0242	0.0291	0.0016	$AFPF$	10.84	910	5	173
	82	2	0.1669	0.0603	0.0657	0.0388	PF				
	61	10	0.4005	0.0906	0.1427	0.1010	KLT				
	72	7	0.3151	0.0735	0.1109	0.0769	CSH				
Class 2	**97.20**	**0.06**	**0.1133**	**0.0246**	**0.0276**	**0.0018**	$AFPF$	**11.30**	**4534**	**0**	**190**
	83.21	**2.99**	**0.1624**	**0.0636**	**0.0610**	**0.0409**	PF				
	64.79	**8.79**	**0.3990**	**0.1048**	**0.1376**	**0.1019**	KLT				
	72.79	**6.79**	**0.3201**	**0.0841**	**0.1145**	**0.0823**	CSH				
Class 2 of [38]	**94.8**	**0.34**	**0.2337**	**0.0241**	**0.0451**	**0.0028**	**[38]**	**7.24**	**3376**	**0**	**235**

E: experiments, P: precision, $\mu_{d_{gc}}$: mean of d_{gc}, $\mu_{d_{gp}}$: mean of d_{gp}, $\sigma^2_{d_{gc}}$: variance of d_{gc}, $\sigma^2_{d_{gp}}$: variance of d_{gp}, FR: system frame rate, NF: number of frames, $min(\Delta x)$: minimum motion vector length, $max(\Delta x)$: maximum motion vector length, $AFPF$: Adaptive Fuzzy Particle Filter Tracker, KLT: Kanade-Lucas-Tomasi Feature Tracker [47], PF: Particle Filter, CSH: CamShift [46], TM: Tracking Method.

each frame in different positions and various scale sizes everywhere. That gives the tracker more candidate regions. Localization accuracy is thus improved. When localization in both AFPF and PF fail, it is because of the similarity or closeness of the color histogram of the target to other samples that are generated. It is even worse for PF, which generates samples everywhere and not necessarily around detected motion regions or a previous target position. This results in more false positive samples or less precision (smaller P values) than with the AFPF tracker. Indeed, optical flow has an important role in removing this type of false positive sample. In addition, it results in the target being almost always located within $1/6^{th}$ of image diameter from the image center.

Tracking with our camera setup and context requires using features that are not too scale or pose dependent, because large instantaneous changes can occur between two frames. Fig. 8 shows an example of KLT feature detection and tracking in one of the video sequences. Because of the large scale and pose changes, all features may

TABLE V

EXPERIMENTAL RESULTS WITH 320×240 IMAGE RESOLUTION.

E	P (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	TM	$FR(fps)$	NF	$min(\Delta x)$	$max(\Delta x)$
E_{11}	95	0.21	0.1520	0.0213	0.0321	0.0019	$AFPF$	9.68	897	11	203
	79	7	0.1982	0.0723	0.0712	0.0561	PF				
	57	11	0.4527	0.1156	0.1732	0.1127	KLT				
	68	9	0.3354	0.0926	0.1362	0.0908	CSH				
E_{12}	93	0.1	0.1161	0.0117	0.0361	0.0018	$AFPF$	10.47	869	5	154
	81	4.3	0.1811	0.0671	0.0657	0.0542	PF				
	59	12	0.3836	0.1062	0.1422	0.1255	KLT				
	72	6	0.3481	0.0821	0.1097	0.0875	CSH				
E_{13}	96	0.2	0.1017	0.0171	0.0318	0.0021	$AFPF$	9.62	904	3	172
	78	7.2	0.1889	0.0612	0.0743	0.0602	PF				
	61	10	0.4159	0.1136	0.1155	0.1117	KLT				
	70	8	0.3290	0.0892	0.0975	0.0887	CSH				
E_{14}	94	0.09	0.1215	0.0112	0.0377	0.0022	$AFPF$	10.51	888	4	129
	82	4	0.1803	0.0652	0.0689	0.0512	PF				
	60	9	0.3967	0.1477	0.1253	0.1089	KLT				
	71	7	0.3619	0.1155	0.1086	0.0861	CSH				
E_{15}	92	0.08	0.1178	0.0526	0.0372	0.0020	$AFPF$	9.76	893	2	118
	80	6	0.1921	0.0663	0.0759	0.0610	PF				
	62	13	0.4007	0.1104	0.1384	0.1176	KLT				
	73	10	0.3679	0.0960	0.1076	0.0904	CSH				
Class 3	**94.01**	**0.13**	**0.1218**	**0.0228**	**0.0350**	**0.0020**	$AFPF$	**9.98**	**4451**	**3**	**203**
	79.98	**5.71**	**0.0664**	**0.1881**	**0.0713**	**0.0565**	PF				
	59.80	**10.99**	**0.4099**	**0.1187**	**0.1389**	**0.1153**	KLT				
	70.78	**8.01**	**0.3485**	**0.0951**	**0.1181**	**0.0887**	CSH				

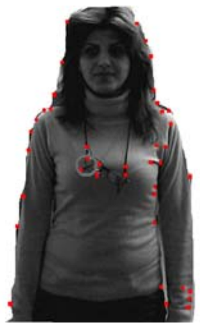
E: experiments, P: precision, $\mu_{d_{gc}}$: mean of d_{gc}, $\mu_{d_{gp}}$: mean of d_{gp}, $\sigma^2_{d_{gc}}$: variance of d_{gc}, $\sigma^2_{d_{gp}}$: variance of d_{gp}, FR: system frame rate, NF: number of frames, $min(\Delta x)$: minimum motion vector length, $max(\Delta x)$: maximum motion vector length, $AFPF$: Adaptive Fuzzy Particle Filter Tracker, KLT: Kanade-Lucas-Tomasi Feature Tracker [47], PF: Particle Filter, CSH: CamShift Tracker [46], TM: Tracking Method.

TABLE VI

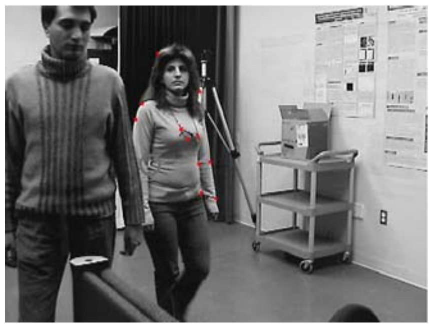
EXPERIMENTAL RESULTS OF AFPF WITH 320×240 IMAGE RESOLUTION FOR TRACKING OF RECYCLING BIN AND BACKPACK.

E	P (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	IMP	$FR(fps)$	NF	$min(\Delta x)$	$max(\Delta x)$	$Object$
E_{16}	95	0.08	0.1124	0.0213	0.0204	0.0014	$Near$	10.72	680	2	140	$Backpack$
E_{17}	94	0.06	0.1156	0.0224	0.0218	0.0016	Far	10.98	674	3	110	$Backpack$
Class 4	**94.5**	**0.07**	**0.1140**	**0.0219**	**0.0211**	**0.0015**	-	**10.84**	**1354**	**2**	**140**	$Backpack$
E_{18}	96	0.07	0.1131	0.0208	0.0237	0.0012	$Near$	11.21	692	1	138	$RecyclingBin$
E_{19}	95	0.06	0.1107	0.0216	0.0252	0.0015	Far	10.87	665	4	159	$RecyclingBin$
Class 5	**95.5**	**0.06**	**0.1119**	**0.0212**	**0.0245**	**0.0014**	-	**11.04**	**1357**	**1**	**159**	$RecyclingBin$

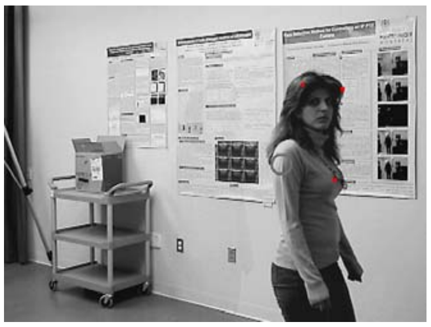
E: experiments, P: precision, $\mu_{d_{gc}}$: mean of d_{gc}, $\mu_{d_{gp}}$: mean of d_{gp}, $\sigma^2_{d_{gc}}$: variance of d_{gc}, $\sigma^2_{d_{gp}}$: variance of d_{gp}, FR: system frame rate, NF: number of frames, $min(\Delta x)$: minimum motion vector length, $max(\Delta x)$: maximum motion vector length, IMP: Initial Model Position from the camera.



(a) $Frame1$



(b) $Frame15$



(c) $Frame16$

Fig. 8. (a) Target introduced to the KLT tracker at $Frame1$, detected and tracked KLT features of target at (b) $Frame15$, and (c) $Frame16$.

be lost between two frames. For example in Fig. 8 (a), there are 40 feature points on the target, while this number is reduced to 13 points in Fig. 8 (b) and to 4 points in Fig. 8 (c). In fact, the appearance changes too much between two images, and all features are often lost since they cannot be tracked. This causes target losses. Although, using color information seems simple, it is better suited to our application because it gives a more generic description.

For the CamShift tracker, the results are explained by the observations made in the introduction. This type of tracker relies on inter-frame proximity of the target in the image plane. When, the camera pans or tilts, or if the frame rate is low, this assumption is no longer valid and results in poor tracking performances. For the PF, we have chosen a sampling scheme that does not rely on proximity. We just sample the image everywhere, with a larger concentration of samples in the middle of the image. This is why our results are superior to those of CamShift. However, the number of samples is not large enough to cover the whole image correctly. Thus, there are tracking failures and large localization errors. Further, the motion detection information is not used to position the samples more efficiently. Putting samples only around the predicted position would not solve this problem because the target may not be near the predicted position. So, in fact, samples should also be around areas with motion, which is the way we are sampling. This experiment shows that a low frame rate and large displacements in the image plane need to be considered explicitly, as proposed by our methodology.

Results show that our algorithm can handle and overcome large displacements in the image plane (i.e. high values of $max(\Delta x)$) and partial occlusion because of a faster frame rate. When $TF \neq 0$, the tracker has lost the target, because of sudden changes of in the target motion direction and the quick walk of the target in the opposite direction. Neither of these situations is predicted by the position predictor. However, we can handle random motion between frames, as long as the target position is well enough predicted to keep the target inside the FOV and its appearance does not change significantly.

With smaller image resolution, the tracking results are significantly better. Indeed, it leads to a faster system rate because fewer data are processed and transferred on the network (for the same JPEG compression level). In fact, because of the larger image size, the camera sends only 16fps in 640×480, while it is capable of sending 30fps in 320×240. If the camera sends only 16fps, delays τ_1 and τ_3 increase. This means that a moving target will have greater motion between two frames than for 320×240. In turn, because the target have greater motion, the camera needs to move through a larger angular distance (keeping its motor operating longer), and thus, delay τ_2 also increases. Because delays τ_1 and τ_2 both increase (τ_2 being the more significant of the two), a larger image results in more localization and prediction errors. This is confirmed by comparing the three classes, where the mean d_{gc}, mean d_{gp}, and P are improved with a smaller image. Similarly, SNC-RZ50N has better performance because of a higher maximum pan-tilt speed than SNC-RZ25N.

Our method is adapted to track other targets as well. This is shown in the Table VI results, and this can be explained by the features used for the target model, which are color features that are adapted to many object tracking problems.

In all our experiments, there are scale changes to verify tracking against scaling. Our algorithm can overcome scaling variations, even in the image with the minimum 5×5 face size (e.g. Fig. 6 (f) and (h)). It can do this because it uses normalized-color histograms and average color features. These two features are relative to the size of the target.

Our method can also recover the tracking if it loses the object (e.g. experiments with $TF \neq 0$), because of the samples around regions with detected motion. Of course, this is conditional on the object being in the FOV of the camera. Occlusions are handled in the same way. However, when the object is occluded, any other similar object will be tracked (the most likely candidate blob) until the occlusion ends. This could cause the real target to fall outside the FOV of the camera, although the camera does not move if the tracked target is not similar enough. Fig. 6 shows an example of the way in which short-term occlusion is handled, and which our proposed method can handle in this case. In the reported experiments, occlusions did not cause difficulties for scenes that are not crowded or for people wearing distinctive clothing (e.g. Fig. 6 (g) and (h)). The duration of the experiments is short, because the goal of the algorithm is to zoom in on the target face and capture it for identification purpose.

E. Study of system limits

In this section, some experiments are discussed to illustrates the limits of the tracking and camera control system. The SNC-RZ50N with 320×240 image size resolution is used. The system has been tested on wide areas of various floors of the Pavillon Mackay-Lassonde at the École Polytechnique de Montréal. We conducted 9 experiments with

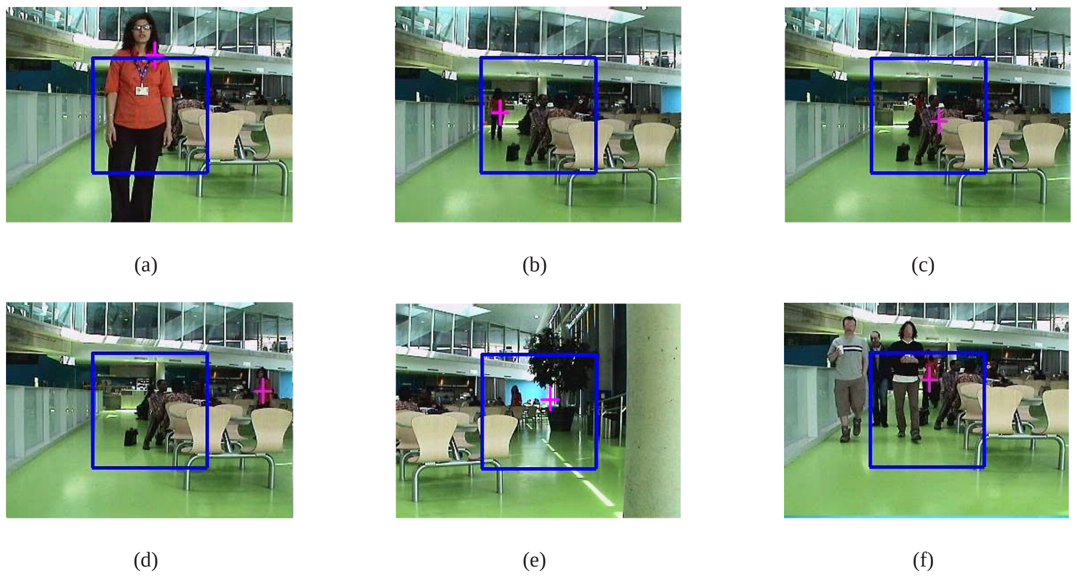


Fig. 9. Some tracking frames in a wide scene, (a) initial target detection, (b) target moves far from the camera, (c) target loss, (d) tracking recovery, (e) target loss, (f) target recovery with partial occlusion. . The blue rectangle has a size equal to $1/3^{th}$ of the image diagonal.

three different actors. The purpose of the following experiments is to determine whether the target is lost because of target size as a result of being far away from the camera, the tracking of similar color objects, or target speed. The actor walks into the scene and moves away from the camera until the camera loses track of him. The size of the target sample where the camera loses the target is assumed to be the smallest target that can be tracked by our system. On average, this target size is at a distance of about 12 meters from the camera. Figure 4.5 shows the frames where the camera has lost the target because of small target size (14×25 pixels in Fig.9 (c), 16×24 pixels in Fig.9 (e)). These target losses are explained by the poor quality of the target color at these sizes, which is not distinctive, and in fact close to black. Also shown in this figure are the frames where the system recovers the track (Fig.9 (d) and (f)). Because the target is larger and its colors are distinguishable, the track is recovered even if the target is occluded by other people .

Fig. 10 shows the average precision P versus target sample size in various experiments for 320 $\times$ 240 images. On average, when the target sample area is smaller than 600 pixels, the detection precision is less than 35$\%$. Similarly, the detection precision is less than 60$\%$ for a target area smaller than 1,000 pixels. The precision increases as the target area increases. However, when the target is too large, the detection precision decreases, because the FOV of the camera is filled with the target, and a small movement of the target results in the whole target, or a portion of it, being outside of the camera's FOV.

In the last set of experiments, we calculated the average maximum target speed at which the system can track the target without losing it versus the target size in pixels and the system frame rate. We conducted 24 experiments classified in 6 classes. Each class belongs to a specified target size. In these experiments, an actor has been asked to walk at various speed on a straight line at different distances from the camera but parallel to the image plane

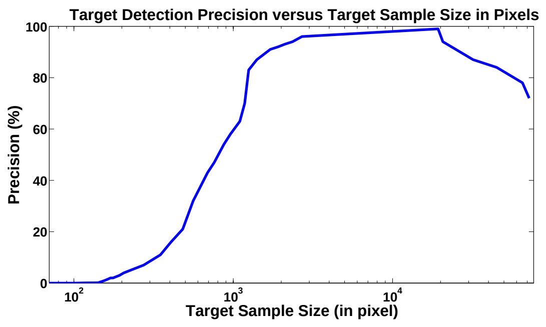


Fig. 10. Average target detection precision versus target sample size in pixels.

TABLE VII

AVERAGE MAXIMUM TARGET SPEED TRACKED BY THE SYSTEM VERSUS TARGET SIZE, TARGET DISTANCE FROM CAMERA AND SYSTEM FRAME RATE.

Experiment	C_1	C_2	C_3	C_4	C_5	C_6
Average Target Size (pixels)	15×45	20×60	30×90	40×120	70×210	80×240
Average Target Distance (m)	10	8	5.5	4.2	2.3	1.5
Average Target Speed (m/s)	2.5	1.9	1.5	1.1	0.8	0.5
Average Frame Rate (fps)	12.3	12.2	12.6	12.5	12.1	12.7

from right to left or left to right. We asked each actor to use a chronometer to record the time needed to walk a specific linear path (5 meters in length). The image size is 320 $\times$ 240.

Table VII shows the average maximum target speed that can be tracked by our system versus target size. The average target size, target distance from the camera, average target speed, and average frame rate in each class are calculated. The average target speed is computed by dividing the distance walked by the total time. The average normal speed of a human is about 4.51 km/h, which is 1.25 m/s [48]. The system can track a target moving at a higher speed when it is smaller and far away from the camera. On the contrary, when the target is close to the camera, since the camera's FOV is limited, the camera needs more time to center on the target. Therefore, a lower maximum target speed is obtained.

VI. CONCLUSION

We have proposed an adaptive fuzzy particle filter (AFPF) method adapted to general object tracking with an IP PTZ camera. Particle filter samples are weighted using fuzzy membership functions. In our particle filter, target modeling and tracking are achieved based on sampling around a predicted position obtained by a position predictor, and moving regions detected by optical flow. Geometric and appearance features of samples are compared to find the most similar target sample among the samples generated by the particle filter. The sample features are scored based on fuzzy rules.

Results show that our algorithm can handle and overcome large displacements in the image plane between two

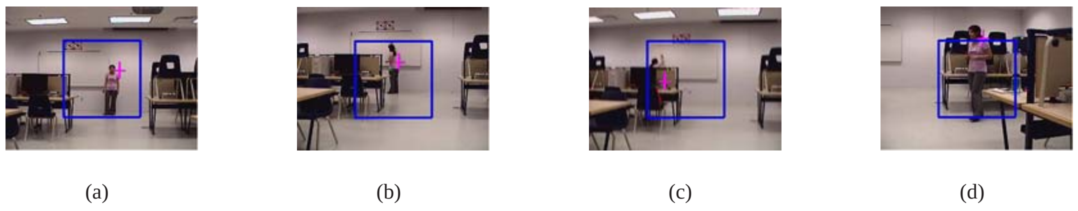


Fig. 11. Example of a classroom application. The blue square delimits the area where we wish the target to be. (a) initial model, (b) before occlusion, (c) after occlusion, (d) scale variation.

consecutive frames, and the detected target location is near to the ground-truth. Also, the camera can center on the target with a good precision. This is because particle filter samples are distributed well in each frame with various scale sizes around candidate locations. The proposed method can handle short-term occlusion on condition that the object stays in the FOV. We will lose a target if the person changes his direction of motion suddenly and walks very quickly in the opposite of the predicted direction. We can recover the track if the target moves back into the camera's FOV. As such, in a distributed network of cameras, the target model features could be propagated to the other cameras. We get better results with a smaller image, because the system delays are reduced resulting in a faster frame rate and in turn smaller displacements in the image plane. We have also shown that our tracking system can be applied to track other objects for different applications. The proposed method can be applied in various videosurveillance applications, including video conferencing and tracking a speaker during a presentation. Fig. 11 shows an example of a teacher tracked in a classroom. To zoom on the face of a speaker and recognize him, in future work we will integrate zooming inside our tracking framework by evaluating the quality of samples at each frame. Good samples means that we have a good tracking. In this case, the camera could zoom on the target.

VII. Acknowledgment

This work is supported by the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT), by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Canada Foundation for Innovation (CFI). We would like to thank Cynthia Orman for revising the paper.

References

[1] N. Bellotto, E. Sommerlade, B. Benfold, C. Bibby, I. Reid, D. Roth, C. Fernández, L. V. Gool, and J. Gonzàlez, "A distributed camera system for multi-resolution surveillance," *Proc. of the 3rd ACM/IEEE Int. Conf. on Distributed Smart Cameras (ICDSC)*, 2009.

[2] C. Chen, Y. Yao, C. Jr., B. Abidi, A. Koschan, and M. Abidi, "Heterogeneous fusion of omnidirectional and ptz cameras for multiple object tracking," *IEEE Trans. Circuits Syst. Video Techn*, vol. 18, no. 8, pp. 1052–1063, 2008.

[3] X. Cindy, F. Collange, F. Jurie, and P. Martinet, "Object tracking with a pan-tilt-zoom camera: application to car driving assistance," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1653–1658, 2001.

[4] S. Kang, B. Abidi, and M. Abidi, "integration of color and shape for detecting and tracking security breaches in airports," *International Carnahan Conference on Security Technology*, pp. 289–294, 2004.

[5] N. Krahnstoever, T. Yu, and S. Lim, "Collaborative real-time control of active cameras in large scale surveillance systems," *European Conference on Computer Vision (ECCV)*, 2008.

[6] S.-N. Lim, A. Elgammal, and L. Davis, “Image-based pan-tilt camera control in a multi-camera surveillance environment,” *Proceedings on International Conference on Multimedia and Expo (ICME)*, vol. 1, pp. 645–648, 2003.

[7] I. Everts, N. Sebe, and G. Jones, “Cooperative object tracking with multiple ptz cameras,” *International Conference on Image Analysis and Processing(ICIAP)*, pp. 323–330, 2007.

[8] Y. L. and S. Payandeh, “Cooperative hybrid multi-camera tracking for people surveillance,” *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1365–1368, 2008.

[9] N. Krahnstoever, P. Tu, T. Sebastian, A. Perera, and R. Collins, “Multi-view detection and tracking of travelers and luggage in mass transit environments,” *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance and CVPR*, 2006.

[10] R. Collins, A. Lipton, and T. Kanade, “A system for video surveillance and monitoring,” *Proceedings of the American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems*, 1999.

[11] N. Krahnstoever and P. Mendonca, “Bayesian autocalibration for surveillance,” *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 1858–1865, 2005.

[12] J. H. Elder, S. Prince, Y. Hou, M. Sizintsev, and E. Olevsky, “Pre-attentive and attentive detection of humans in wide-field scenes,” *International Journal of Computer Vision*, vol. 72, no. 1, pp. 47–66, 2007.

[13] T. Funahashi, T. Fujiwara, and H. Koshimizu, “Hierarchical tracking of face, facial parts and their contours with ptz camera,” *IEEE Int. Conf. on Industrial Technology (ICIT)*, vol. 1, pp. 198–203, 2004.

[14] T. Funahashi, M. Tominaga, T. Fujiwara, and H. Koshimizu, “Hierarchical face tracking by using ptz camera,” *IEEE Int. Conf. on Automatic Face and Gesture Recognition (FGR)*, pp. 427–432, 2004.

[15] S. Kang, J. Paik, A. Koschan, B. Abidi, and M. Abidi, “Real-time video tracking using ptz cameras,” pp. 103–111, 2003, 6 th Int. Conf. on Quality Control by Artificial Vision.

[16] Y. Yao, B. Abidi, and M. Abidi, “3d target scale estimation and motion segmentation for size preserving tracking in ptz video,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pp. 130–136, 2006.

[17] T. Matsuyam, S. Hiura, T. Wada, K. Muease, and A. Toshioka, “Dynamic memory: architecture for real time integration of visualperception, camera action, and network communication,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 728–735, 2000.

[18] A. Elgammal, D. Harwood, and L. Davis, “Non-parametric model for background subtraction,” *IEEE workshop on FRAME-RATE*, pp. 751–767, 2000.

[19] M. Roha, T. Kima, J. Park, and S. Lee, “Accurate object contour tracking based on boundary edge selection,” *Pattern Recognition*, vol. 40, no. 3, pp. 931–943, 2007.

[20] R. Venkatesh Babu, P. Perez, and P. Bouthemy, “Robust tracking with motion estimation and local kernel-based color modeling,” *Image and Vision Computing*, vol. 25, no. 8, pp. 1205–1216, 2007.

[21] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE T-PAMI*, vol. 25, no. 5, pp. 564–577, 2003.

[22] W. Qu and D. Schonfeld, “Robust control-based object tracking,” *IEEE Transactions on Image Processing*, vol. 17, pp. 1721–1726, 2008.

[23] A. D. Bagdanov, A. del Bimbo, and W. Nunziati, “Improving evidential quality of surveillance imagery through active face tracking,” *Proc. of International Conference on Pattern Recognition (ICPR)*, pp. 1200–1203, 2006.

[24] P. Viola and J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[25] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, “Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans,” *IEEE T-PAMI*, vol. 30, no. 10, pp. 1728–1740, 2008.

[26] I. Leichter, M. Lindenbaum, and E. Rivlin, “Bittracker- a bitmap tracker for visual tracking under very general conditions,” *IEEE T-PAMI*, vol. 30, no. 9, pp. 1572–1588, 2008.

[27] H. Zhou, M. Taj, and A. Cavallaro, “Target detection and tracking with heterogeneous sensors,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 503–513, 2008.

[28] P. Pan and D. Schonfeld, “Dynamic proposal variance and optimal particle allocation in particle filtering for video tracking,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1268–1279, 2008.

[29] P. Vadakkepat and L. Jing, “Improved particle filter in sensor fusion for tracking randomly moving object,” *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 5, pp. 1823–1832, 2006.

[30] Y. Zhai, M. B.Yeary, and S. Cheng, “An object-tracking algorithm based on multiple-model particle filtering with state partitioning,” *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 5, pp. 1797–1809, 2009.

[31] E. Sommerlade and I. Reid, “Information-theoretic active scene exploration,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7, 2008.

[32] J. Ahmed, M. Jafri, M. Shah, and M. Akbar, “Real-time edge-enhanced dynamic correlation and predictive open-loop car-following control for robust tracking,” *Journal of Machine Vision and Applications*, vol. 19, no. 1, pp. 1–25, 2008.

[33] C. Micheloni and G. L. Foresti, “Zoom on target while tracking,” *IEEE international conference on image processing*, vol. 3, pp. 117–120, 2005.

[34] J. Shi and C. Tomasi, “Good features to track,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, 1994.

[35] C. Tomasi and T. Kanade, “Detection and tracking of point features,” *Carnegie Mellon University Technical Report*, vol. CMU-CS, pp. 91–132, 1991.

[36] D. Schreiber, “Generalizing the lucas-kanade algorithm for histogram-based tracking,” *Pattern Recognition Letters*, vol. 29, no. 7, pp. 852–861, 2008.

[37] H. Zhou, Y. Yuan, and C. Shi, “Object tracking using sift features and mean shift,” *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[38] P. Darvish Zadeh Varcheie and G.-A. Bilodeau, “Active people tracking by a ptz camera in ip surveillance system,” *IEEE International workshop on Robotic and Sensors Environments (ROSE)*, 2009.

[39] A. Torabi and G.-A. Bilodeau, “Measuring an animal body temperature in thermographic video using particle filter tracking,” *Lecture Notes in Computer Science: Advances in Visual Computing*, vol. 5358, pp. 1081–1091, 2009.

[40] N. Bellotto and H. Huosheng, “People tracking and identification with a mobile robot,” 2007, iEEE Int. Conf. on Mechatronics and Automation (ICMA).

[41] Math Forum, “Points within an ellipse,” 2003, http://mathforum.org/library/drmath/view/63045.html, [Online; accessed 1-April-2009].

[42] S. H. Cha and S. N. Srihari, “On measuring the distance between histograms,” *Pattern Recognition*, vol. 35, no. 6, pp. 1355–1370, 2002.

[43] Sony corporation, “Snc-rz25n/p cgi command manual,” 2005, version 1.0.

[44] F. Yin, D. Makris, and S. Velastin, “Performance evaluation of object tracking algorithms,” 2007, iEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance(PETS).

[45] M. Isard and A. Blake, “Condensation - conditional density propagation for visual tracking,” *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[46] Intel Corporation, “Camshift tracker,” 2001, http://worldlibrary.net/eBooks/Give-Away/Technical_ eBooks/OpenCVReferenceManual.pdf, [Online; accessed 21-May-2010].

[47] S. Birchfield, “Kanade-lucas-tomasi feature tracker,” 1997, http://www.ces.clemson.edu/~stb/klt/, [Online; accessed 27-November-2009].

[48] Wikipedia, “Walking — wikipedia, the free encyclopedia,” 2010, http://en.wikipedia.org/wiki/Walking#cite_ note-1, [Online; accessed 17-May-2010].