Machine vision and applications manuscript No.
(will be inserted by the editor)

People Tracking Using a Network-based PTZ Camera

Parisa Darvish Zadeh Varcheie · **Guillaume-Alexandre Bilodeau**

Received: date / Accepted: date

Abstract In this paper, we propose a method for online upper body tracking using an IP PTZ camera. This type of camera uses a built-in web server resulting in variable response times when sending control commands. Furthermore, communicating with a web server implies network delays. Thus, because the camera is inside a control loop, the effective frame rate that can be processed by a computer vision method is irregular and in general low (2-6 fps). Our tracking method has been designed specifically to perform in such conditions. It detects at every frame, candidate blobs using motion detection, region sampling, and region color appearance. The target is detected among candidate blobs using a fuzzy classifier. Then, a movement command is sent to the camera using the target position and speed. The proposed method can cope with low frame rate and thus with large motion of the target even in the case of fast walk. Results show that our system has a good target detection precision ($> 88\%$), low track fragmentation, and the target is almost always localized within 1/6th of the image diagonal from the image center.

Keywords Fuzzy tracking · Feature-based tracking · People tracking · Low frame rate tracking · IP PTZ camera

1 Introduction

Object tracking is an active subject and has a lot of applications in the field of computer vision. Here, object tracking is applied to human upper body tracking. Upper body tracking determines the location of the upper body for each input image of a video. It can be used to get images of the face of a human target entering a perimeter or to follow a speaker in a classroom or at a conference. 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 command via its integrated web server. It allows a distributed access from internet (access from everywhere, but with non-defined delay). Tracking with such camera implies the following challenges:

Department of Computer Engineering and Software Engineering,
École Polytechnique de Montréal,
P.O. Box 6079, Station Centre-ville Montréal (Québec), Canada, H3C 3A7
E-mail: {parisa.darvish-zadeh-varcheie, guillaume-alexandre.bilodeau}@polymtl.ca

(a) (b) (c)

Fig. 1 Camshift tracking failure. Track with Camshift (red ellipse) and track with our proposed method (pink cross) for frames of E_7, (a) initial model selection for Camshift (b) and (c) short-term occlusion and camera panning.

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

To illustrate some of the problems we are facing with such a network camera, we applied the Camshift method for tracking the face of a person [8]. Fig. 1 shows a tracking failure for two consecutive frames. This failure comes from processing a video with 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 near the previous target position. In such a case, methods like mean-shift cannot converge toward the target, because it gets trapped inside a local maximum. Thus, instantaneous large displacement of the target needs to be accounted for explicitely. This is what we want to show in this paper by proposing an adapted tracking framework.

Our framework accounts for delays and large motion in its different processing steps. For each step, different methods may be used if they respect some constraints related to low frame rate. In this paper, we propose simple and efficient ones. We detect, at every frame, candidate blobs using motion detection, region sampling, and region color appearance. The target is detected among candidate blobs using a fuzzy classifier. Then, a movement command is sent to the camera using the target position and speed. When tracking is done with high confidence, the camera zooms on the target. Although applied to track human upper body, our propose 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 it accounts for camera response time and networks delays proper to such camera. It can be extended to other features or targets by adapting the target model and sampling scheme.

Results show that our upper body tracking system has a good target detection precision ($> 88\%$), low track fragmentation, and the target is usually localized within 1/6th of the image diagonal from the image center. In addition, results show that our proposed framework can cope better than the state-of-the-art with large motion of the target by considering network delays and camera response time.

Section 2 gives an overview of the state-of-the-art. Section 3 describes the servo system architecture. Section 4 presents our proposed upper body tracking algorithm, and section 5 gives validating experiments. Section 6 concludes the paper.

2 Related works

Our main problem to solve is object localization under large target motion between two consecutive frames and with backgound motion. Most works on tracking with PTZ camera do not account for both of these difficulties. There are three families of works: 1) Tracking with a PTZ camera and a static camera; in this case, the difficulty related to background motion is 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, background motion is 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; this is the most difficult case, and in general low frame rate is not accounted for. In fact, most works account for background motion, but not for low frame rate, and methods that consider low frame rate, do not account for background motion. Here, we must account for both.

2.1 Tracking with a PTZ camera and static camera

Several works have been introduced with combination of a PTZ camera with multiple PTZ or fixed cameras in a master-slave configuration to explore the field of view [5,11,12,24, 28,32]. The important part in the combination of a PTZ camera with other PTZ cameras or stationary cameras is finding the geometrical relationship between them. For this purpose, camera calibration is an appropriate solution which is applied in [5,18,29]. In these typical works, a system contains distributed static and PTZ cameras that are controlled by visual tracking algorithms with a central supervisor unit. There is a calibration between the cameras to know the geometrical relations between them. The moving object detection and tracking is done by static camera, and the PTZ camera is commanded to follow the target from the tracking results of the static camera. For example, Krahnstoever *et al.* [28] designed a real-time control system of active cameras for a multiple camera surveillance system. Object tracking is done by fixed cameras using a shape-based method [27]. The cameras are calibrated using a common site-wide metric coordinate system [13,26]. The target coordinates are transformed to the appropriate pan and tilt values using geometrical transformation, and the camera is moved accordingly. Further, Elder *et al.* [16] and Funahasahi *et al.* [19,20] proposed similar approach using face detection methods.

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 and the background motion problem is solved by the static camera. The PTZ camera is passive and just moves based on the position of the target in other camera's field of view. Thus it is a problem different from what we are studying, where the PTZ camera must track a target without any other information than the one it can extract from its own field of view. The advantage of object tracking using only one PTZ camera is removing the requirement of fixed or stationary cameras to reduce the cost of the surveillance system, while keeping a large coverage of the scene.

2.2 Tracking with a PTZ camera using background subtraction

Another approach is to solve the background motion problem by learning the background for all camera positions and zooms [32,17,25,35]. Background subtraction may then be applied for object localization. Then, standard static camera approaches may be used like edge matching and Kalman filtering [17], edge contours matching [36,46,2], or dynamic memory methods [35,45].

In these methods, the background motion problem is solved at the expense of a complex initialisation and background updating computations. If the camera is moved, the learned backgrounds are not valid. This approach thus lacks flexibility.

2.3 Tracking only with a PTZ camera

In this approach, the tracking method must be chosen to account for non-static background. To help tracking, it is implicitly assumed that the motion is small between frames (see Section 1). Thus, methods such as edge-based tracking with optical flow [37], mean-shift [43, 14,8], particle filter [22], Viola and Jones face detector [44] combined with Shi-Tomasi feature point tracker [3,10,39,42], KLT [6], histogram-based tracking [38] and edge and Kalman filtering [1,40] are used.

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

2.4 Tracking under low frame rate

Li *et al.* [31] developed a cascade particle filter method with discriminative observers of different life spans for low frame rate videos. Each observer or observation model (such as 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 increases computation time. The method has limitations in distinguishing between different targets, and has model over updating problems. Recently, Leichter *et al.* [30] 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 pixel 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.

Our proposed method is related to these methods as our aim is to maximize similarity of target appearance in consecutive frames without strong consideration on spatial proximity, but in addition, it considers a PTZ camera and thus requires motion prediction and robustness to background changes. In our work, we want to cope with the problem of large motion of targets, low usable frame rate, background changes, and tracking with various scale changes. In addition, the tracking algorithm should handle the camera response time and zooming.

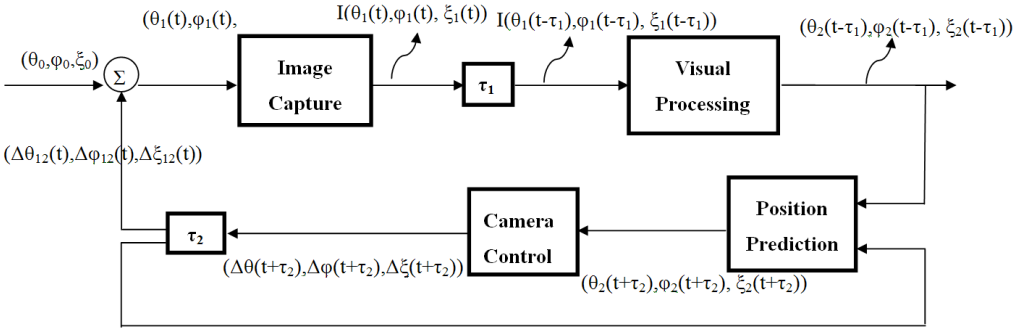


Fig. 2 The system architecture and servo control model. (θ_0,ϕ_0, ξ_0):initial pan-tilt angles and zoom value,($\Delta\theta_{12}$,$\Delta\phi_{12}$, $\Delta\xi_{12}$) means (θ_1-θ_2,ϕ_1-ϕ_2,$\xi_1 - \xi_2$), and I($\theta_i(t)$,$\phi_i(t)$,$\xi_i(t)$) means image at time t for angles (θ_i,ϕ_i) and zoom (ξ_i).

3 System architecture and method overview

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, and dynamic. These properties help us in modeling of the system. Indeed, we can use the characteristics of each category to represent our system model. For example, the system is causal, and thus it means that our system response at time t depends only on values occurring now or that occurred in the past. This causality information helps us to predict the current situation. Also the system is discrete which means that we do not capture continuously and there is a delay between each image. This is modeled by a delay block τ_1. The system is stable which means that for finite input (pan-tilt-zoom values), there is finite output (obtained pan-tilt-zoom values). The system is obviously time variant and varies according to the scenario, processing time, network traffic delay, and other events that are happening. It is a dynamic system that is constantly changing because its value depends on the present value of input and past values.

The servo control and tracking system is modeled by a closed-loop control which has a negative feedback as shown in Fig. 2. To design the servo control system, we selected its components based on background motion and low frame rate assumptions. Thus, in addition to image capture and camera control, we need a visual processing component that can track an object having large motion between consecutive frames because of image capture delay τ_1, and a position predictor that moves the camera based on delay τ_1 and a delay τ_2 (delay to change the camera position or zoom) to compensate them to keep the target in the field of view (FOV). τ_1 is almost constant and is about 0.05s for 640×480 image. τ_2 varies in the range of [0.71s, 1.5s] depending on the pan, tilt and zoom values, and network traffic. The input of the system is the current pan and tilt angles and zoom value of the camera and the output is the determined pan and tilt angles and zoom value by the visual processing.

For visual processing, the chosen method must not rely on target proximity between frames for tracking. Thus, we have chosen a method that samples regions in a frame based on motion and around image center (the likely target positions), and then, the target is localized using color appearance. Thus, large, but likely parts of the image, are used as search windows. For position prediction, we just need to estimate the motion of the target during the delays and position the camera accordingly.

To allow the visual processing component to maintain control of the camera motion, camera movement and image analysis are performed successively and not simultaneously.

Because of latency of the network and of the camera response, it is not possible to control the camera continuously. Camera commands are queued and performed one after the other. If a new command is sent and the queue is not empty, most probably this command will be applied too late, and target will be lost. In our servo control system, a new motion command is issued only if the queue is empty. By this way, good control of the camera is maintained, but the target may move during the motion command. Thus, the target motion needs to be estimated to keep the target inside the FOV of the camera.

4 Methodology

The algorithms that are used for visual processing, position prediction and camera control are explained in the following. First, we have made the following assumptions for our application of human upper body tracking:

- We consider that persons walk at a normal pace or fast, but do not run.
- The target person can walk in any direction, but the face should be always visible partially.
- The target person has distinctive colors.
- We assume an initially wide FOV (approximately $48\,^{\circ}$) and scenes that are not crowded (max 2-3 persons).

4.1 Visual processing

4.1.1 Target modeling

To represent the target, it is first required to model it by some features, like edges, feature points, or color. Since we are working in a low frame rate context, under about constant illumination, but with variability in the background because of camera pan and tilt changes, with scaling variations, and with large target motion displacements, we need a feature which is stable enough, scale invariant, and discriminative. Although simple, a color-based model satisfies these requirements. Other models could be used with our methodology if more precise details need to be accounted for. However, the model must be robust to instantaneous large scale and viewpoint change because of low frame rate. This is coherent with the rational of the work of [30], which also deals with low frame rate.

For example, methods like Dalal *et al.* [15] seems appealing to detect a person. In their algorithm, a dense grid of Histograms of Oriented Gradients (HOG) is calculated over blocks of 16 $\times$ 16 pixels used as detection window. A linear SVM classifier is used to classify the human. This method is robust enough but it can only process images with 320 $\times$ 240 size at a low frame rate ($1fps$). Zhu *et al.* [48] have speeded up the algorithm of [15] to process 320 $\times$ 240 size images at $5fps$. The problem with both methods is high computational cost. Also, HOG is sensitive to the viewpoint of the person, arms and legs position. The model should be learned for all possible viewpoints and poses of the human. In addition, it is required to process all edges or contours inside the whole image in each frame to extract the body edge.

Therefore, our target is represented by an elliptical image region modelled by its color content. It is modeled by two features: 1) normalized quantized HSV color histogram and 2) average RGB color values. Histogram quantization is used to reduce computation time and for an adequate level of precision; we use an 18 x 3 x 3 = 162 bins color histogram. The

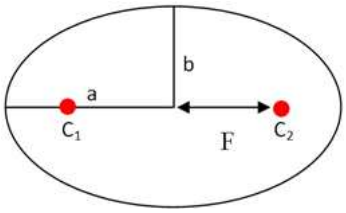


Fig. 3 The relationship of ellipse and the foci.

histogram is normalized. The second feature is the mean of R, G and B color components of RGB color space of all the pixels inside of the elliptical region.

4.1.2 Track initialization

To track the target, its initial position and size are first determined. Currently, initialization is done manually. It is performed by selecting the top part of the body (head and torso) of the person to track. It is a part that should always be visible, either when the person is far away or when it is close to the camera. The torso is 1/3 of the total height of the body as considered in Bellotto *et al.* [4]. We fit an ellipse inside the bounding box of the selected region, and model the resulting elliptical region with the features of the previous section. This is the initial model M. Fig. 6 (a) and (e) show elliptical regions (torso and the head). We use an elliptical region because it fits better the shape of the head and torso. To extract the pixels inside of the elliptical region from the selected rectangular region, a mathematical rule is applied [33]. Let C_1 and C_2 be the two focus points of an ellipse (foci of an ellipse) and a and b be the lengths of the ellipse major and minor axis respectively, obtained from the rectangular region. According to Fig. 3, the location of foci is obtained using [34]:

$$F = \sqrt{a^2 - b^2}, \tag{1}$$

F is the distance of the foci from ellipse center. A point P is inside of an ellipse if the sum of its distances from the two focus points is smaller than the length of major axis:

$$|P - C_1| + |P - C_2| < 2a \tag{2}$$

The next step is to find candidate elliptical regions to localize the target in the next image.

4.1.3 Candidate elliptical regions detection

Because of background motion and low frame rate, the target may move over a large distance in the image plane between two video frames. Thus, the search area for target localization must not solely rely on proximity with the previous position. The target can be at two likely positions in the image: 1) if the target stops moving, because the camera centers on it, the target will be around the center of the image, and 2) if the target is moving, its position can be determined with frame differencing. Thus, to localize the target, we just need to search around the center of the image and around regions with motion. Localization is done by sampling with ellipses around the likely target position and then comparing color appearance with the target model. Candidate blobs are detected as described in the following sections.

4.1.4 Motion-based candidate elliptical regions

In our application, the PTZ camera may move. Thus, to detect the motion, we do the difference of two consecutive frames when the camera is still [36,46]. Recall that when the camera is moving, no processing is performed because the camera response time and network delays do not allow accurate continuous control of the camera. So it is better to wait that it has stopped moving before moving it again. It is a compromise, but it allows us to keep control of the camera motion.

Because image difference results are noisy, some morphological operations such as erosion, dilation, filtering and image closing are used to reduce noise. Detected motion pixels are grouped into regions using connected components. Because of image differencing, whenever a moving object in the scene has a color similar to the background or has an overlap with its previous frame position, some parts of the moving object are not detected as foreground regions. This results in detecting smaller blobs that are fragments of a larger one. Fragments are merged iteratively based on their proximity. The small regions that are nearby, and whose contours are in intersection, are merged. This process is done until there is no blob that intersects with others.

For sampling the image at these areas, ellipses are positionned and sized on each blob based on skin location in the blob. Alternatively to this method for sampling motion blobs, the motion areas could be sampled using ellipses with random positions and sizes. In our application, it was more efficient to use skin information. Thus, we apply Bayesian skin classification over the candidate blobs to filter outliers and to determine ellipse position and size. For sizing the ellipse and for their modeling, the torso is assumed to be below the detected skin region. The torso is two times longer than the height of detected skin region. Thus, the ellipse width is the same as the skin region width, and its height is three times longer than the skin region height. This assumption is based on the human head and torso location. The ellipse is then modeled as in Section 4.1.1.

We selected a Bayesian skin classifier because of its high true positive rate, simple implementation and minimum cost [23]. The decision rule in Bayesian skin classifier can be expressed as:

$$\begin{cases} \text{if } \frac{p(x|\omega_1)}{p(x|\omega_2)} > T_s & skin \\ \text{else} & nonskin. \end{cases}$$

where $p(x|\omega_1)$ and $p(x|\omega_2)$ are the class-conditional probability density functions of skin and non-skin color respectively. They are estimated in an off-line process by a non-parametric density estimation method. From a large number of labeled skin and non-skin pixels in a training data set, normalized skin and non-skin histograms are calculated. These two histograms are used as the class-conditional probability density functions of skin and non-skin color. T_s represents the adjustable threshold (here $\tau = 0$).

After training, the classifier is used in the online process. We remove all the skin regions that contain less than 40 skin pixels (less than the half of the minimum face size). Then, the candidate blobs that do not contain skin region or that are too small based on the maximum size of the extracted blobs in the current frame are removed. Then sampling is applied on the remaining blobs.

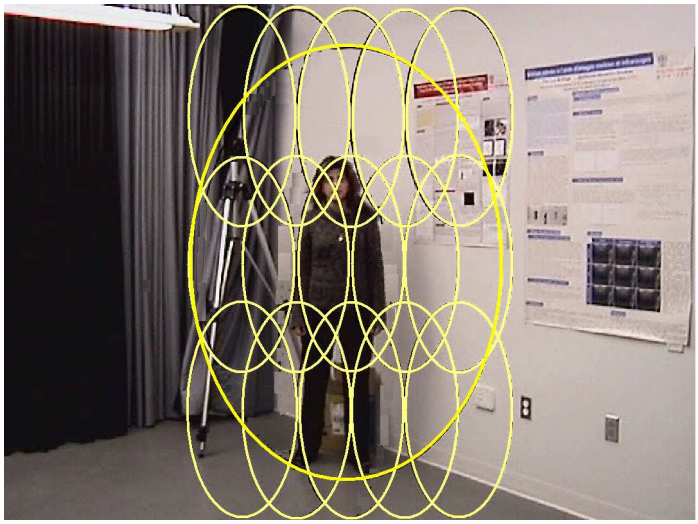


Fig. 4 Sixteen fixed elliptical candidate blobs considered in each frame.

4.1.5 Fixed (regions elliptiques fixes (fixes dans le sens, toujours la meme position), en franais) candidate elliptical regions

According to our goal, the object should be always near the image center. To have robust tracking even when there is no motion from the target, we consider F additional fixed candidate blobs in the image (typically $F = 16$). Fig. 4 shows the location of these fixed candidate blobs. The largest ellipse is used for zooming or object approaching the camera. Its area is 1/3 of the image area. The small ellipses are used for a target far from the camera and close to the center in different positions. These fixed candidate blobs are located around the image center, because the camera should be close to be centered on the target when it stops moving. The sizes of these ellipses are obtained experimentally according to the minimum face size, which is in our algorithm 5x5 pixels from frontal and lateral views.

In this case, skin classification is used only to remove the candidate region with less than 40 skin pixels.

4.1.6 Target localization using a fuzzy classifier

At this step, we have many candidate elliptical regions. We must determine which one is more likely to be the target. Thus, features of candidate elliptical regions ER_i are compared with the initial model M, and a score ($ScoreER_i$) is given to each ER_i using a fuzzy rule. A fuzzy approach allows us more flexibility for determining $ScoreER_i$. Normalization of $ScoreER_i$ might be done by any function. For example, the camera is always moved to center on the target. This results in target location probability to be higher around the image center. Thus, in that case, a Gaussian membership function is an appropriate choice. Fuzzy membership functions are applied to geometric and appearance features to normalize them. The target is the candidate elliptical region with the highest score. In the following, we describe the fuzzy inputs (e.g. distances) and fuzzy membership functions used in our classifier. The goal of our fuzzy classifier is to define the most discriminating measures to compare robustly the target features with candidate elliptical region features. We apply

different measures on color because each of them addresses different level of accuracy. We want to balance the precision of measures in general conditions. Our appearance based measures, used often in many pattern recognition and image processing applications, are described in the following:

1. Euclidean distance (d_{EC}) of mean RGB of ER_i (R_{eri}, G_{eri}, B_{eri}) with the mean RGB of M (R_m,G_m, B_m). It is defined as

$$d_{EC} = \sqrt{(R_{eri} - R_m)^2 + (G_{eri} - G_m)^2 + (B_{eri} - B_m)^2}. \quad (3)$$

   This distance is between 0 and 255. This distance is a comparison of the average color content of each ER_i with M to verify their similarity. The membership function μ_{EC} for this distance is a linear function and is defined as

$$\mu_{EC}(d_{EC}) = 1 - \frac{d_{EC}}{255\sqrt{3}}. \quad (4)$$

2. Euclidean distance (d_{EP}) of ER_i centroid from the image center. It is defined as

$$d_{EP} = \sqrt{(x_{eri} - x_{im})^2 + (y_{eri} - y_{im})^2}, \quad (5)$$

   where (x_{eri}, y_{eri}) and (x_{im}, y_{im}) are respectively the coordinate vector of the centroid of ER_i and of the image center. The tracker commands the camera to center on the target. Thus, normally, the person should be near the image center. This distance, a geometric feature, is an appropriate measure to evaluate ER_i location according to our goal (i.e camera center should be always on the target based on the previous camera motion). To account for this, the membership function μ_{EP} is defined as a normalized Gaussian function with a peak at the image center. σ^2 is equal to a quarter of the image area around the image center:

$$\mu_{EP}(d_{EP}) = exp(-\frac{(d_{EP})^2}{2\sigma^2}). \quad (6)$$

3. Euclidean distance (d_{EH}) of normalized quantized HSV color histogram of ER_i with the histogram of M. It is computed as [9]

$$d_{EH}(H_{eri}, H_m) = \sqrt{\sum_n (H_{eri}[n] - H_m[n])^2}, \quad (7)$$

   where H_{eri} and H_m denote the normalized histograms of ER_i and M respectively and n is the histogram bin number. The color histogram of a region represents the color distribution of that region. Color histogram comparison is another measure used to compare the appearance closeness of ER_i with M by comparing their color distributions. The membership function (μ_{EH}) that is used for this distance is defined as

$$\mu_{EH}(d_{EH}) = 1 - \frac{d_{EH}}{\sqrt{2}}. \quad (8)$$

4. Similarity (S_H) of normalized quantized HSV color histogram of ER_i with histogram of M [7]. It is the normalized correlation coefficient of two histograms and it is defined as

$$S_H(H_{eri}, H_m) = \frac{\sum_n ((H_{eri}[n] - \bar{H}_{eri})(H_m[n] - \bar{H}_m))}{\sqrt{\sum_n (H_{eri}[n] - \bar{H}_{eri})^2}\sqrt{\sum_n (H_m[n] - \bar{H}_m)^2}}, \quad (9)$$

where $\bar{H}_{eri}$ and $\bar{H}_m$ denote the average of normalized histograms of ER_i and M. (S_H) has a value between -1 and 1. It is used to find the correlation of two color distributions which is a mathematical representation of appearance similarity. This is our most precise color measure. For the membership value μ_H, it is normalized using

$$\mu_H(S_H) = \frac{1 + S_H}{2}. \tag{10}$$

The final score for ER_i $(ScoreER_i)$, is the intersection of all fuzzy sets and it is obtained by multiplying all membership functions. That is

$$ScoreER_i = \mu_{EC}.\mu_{EP}.\mu_{EH}.\mu_H. \tag{11}$$

In each frame, all ER_i are scored using the fuzzy classifier. The score is the similarity likelihood with M. The tracked target is the one with the largest likelihood, that is

$$S_f = max_i(\forall ScoreER_i). \tag{12}$$

4.2 Target position prediction

As discussed in Section 3 and 4.1.4, the delays in the control loop mean that the system cannot center the camera on the target correctly using only the determined object position. To compensate for motion of the target during the delays, a position predictor based on the two last motion vectors has been designed. This motion predictor will consider the angle between two consecutive motion vectors. If the angle difference is smaller than $25\,^\circ$, it is assumed the target is moving in the same direction and its average speed can be estimated. The average speed ($\bar{V}$) of the target for the two last target motion vectors is thus equal to:

$$\bar{V} = \frac{\Delta x_1 + \Delta x_2}{\tau_1^1 + \tau_1^2}. \tag{13}$$

where Δx_1 and Δx_2 are the two target displacement vectors (i.e. target motion vectors). Motion vectors are calculated when the camera is not moving. τ_1^1, τ_1^2 are delay τ_1 between the two last captured images. Thus, the system will put the camera center on the predicted position which is :

$$x_P = x_E + \bar{\tau}_2 \times \bar{V}. \tag{14}$$

where x_P is the predicted target coordinate and x_E is the extracted target coordinate from the fuzzy classifier. $\bar{\tau}_2$ is the average delay time τ_2 obtained from previous camera movements. Thus the camera is controlled and moved adaptively with the speed of the network and camera response time, and the displacement vector of the target.

4.3 Camera control

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

$$C_x = P_x \times \frac{C_z}{P_z} \tag{15}$$

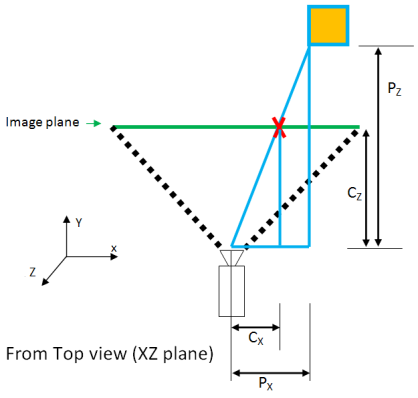


Fig. 5 Perspective transform geometry.

$$C_y = P_y \times \frac{C_z}{P_z}. \tag{16}$$

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.

Zooming is done when the system has a good confidence about tracking results to avoid losing the object from observed camera FOV. Therefore, zooming is done under a criterion on target score similarity S_f. In each frame, the mean μ_{Sf} and variance σ_{Sf} of all S_f from first frame up to current frame are calculated. A zoom in command is sent only if the following criterion is met:

$$S_f > \mu_{Sf} + \sigma_{Sf}. \tag{17}$$

The zoom out command is sent if this inequality holds:

$$S_f < \mu_{Sf} - \sigma_{Sf}. \tag{18}$$

Each zoom command increase by two or decrease by one-half the FOV size. Thus, to follow the target, the PTZ motors are commanded based on image coordinate of x_P and appropriate zoom value. As discussed earlier, in our servo control loop, computing x_P and moving the camera are consecutive (not simultaneous) to keep control over the camera in the case of variable delays. To control the camera, we have implemented two methods. Angles can be computed by the camera using the on-board *AreaZoom* function, or the camera can be controlled by computing the angles on a workstation and sending computed angles to the camera using the the *RelativePanTiltZoom* function. Each of these two ways are explained in the following. Zoom commands are sent independently using the on-board *MultipleZoom* function that zooms m times.

4.3.1 Using the AreaZoom function

The inputs of this function are the target pixel coordinates x_P in the current frame to which the camera should center. *Width* and *Height* are two other inputs that are used for zooming purpose, but we do not use them. Camera is controlled by sending HTTP POST request

Table 1 Cameras specifications.

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

using the CGI scripts of the camera [41]. The speed of motion of the camera is determined automatically based on the required displacement. The larger displacement causes camera moves faster.

4.3.2 Using the RelativePanTiltZoom function

In this case, the angles required to move the camera are computed by the workstation. The target pixel coordinates x_P in the current frame is converted in pan-tilt values for centering the camera. The pan and tilt values are obtained according to the field of view of the lens and the range of pan and tilt angles for each camera (i.e. it is dependent on the camera model). The angles are sent to the camera by sending an HTTP POST request using the CGI scripts of the camera [41] and the *RelativePanTiltZoom* function. For this function, it is possible to specify the speed of motion. It is set to the maximum value. The computations are also faster because they are done by the workstation. Thus, the camera moves faster.

5 Experiments and analysis

5.1 Data acquisition and ground-truth

Our method has been implemented on an Intel Xeon® 5150 in C++ using OpenCV and a custom library to control the IP PTZ cameras. We used two Sony IP PTZ cameras (SNC-RZ50N and SNC-RZ25N) for our tests. Table 1 shows the capabilities of both cameras. 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. Experiments are done in a room with dimension of 6×4 meters. We recorded all the experiments to extract their ground-truth manually for performance evaluation. Only usable frames are recorded (when the camera is not processing a movement command) as these frames are the only one that are processed by our algorithm as explain in section 3 and section 4. 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 from the frontal view is selected for initial modeling. She starts to walk around in a room. Two or three actors can walk at the same time in different directions, crossing and occluding with the target. The target makes some pauses while walking to verify the performance for stationary target. The target actor also moves parallel, toward or away from the camera. Fig. 6 shows the initial model selection and some frames obtained during tracking.

Fig. 6 Examples of tracking frames for E_8 (a) to (d), E_2 (e) to (h), E_{35} (i) to (l), and E_{38} (m) to (p). E_8 (a) initial model selection, (b) short-term occlusion, (c) after occlusion, (d) scale variation; E_2 (e) initial model selection, (f) short-term occlusion, (g) after occlusion, (h) scale variation; E_{35} (i) initial model selection, (j) short-term occlusion, (k) after occlusion, (l) scale variation; E_{38} (m) initial model selection, (n) short-term occlusion, (o) after occlusion, (p) scale variation.

5.2 Evaluation metrics

To evaluate our method, five metrics are used:

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

$$P = \frac{TP}{TP + FP}, \tag{19}$$

   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. Detection precision (DP) to calculate the target face detection accuracy. It is defined as

$$DP = \frac{CDF}{ToF}, \tag{20}$$

   where CDF is the number of frames in which the target face is detected correctly and ToF is total number of frames that contain the target face. We do not consider the false

positive results. We just count the number of frames in which the system could detect the target face correctly. DP is used for this purpose.

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

$$d_{gc} = \frac{\sqrt{(x_c - x_g)^2 + (y_c - y_g)^2}}{a}, \tag{21}$$

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

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

$$d_{gp} = \frac{\sqrt{(x_p - x_g)^2 + (y_p - y_g)^2}}{2a}, \tag{22}$$

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

5. Track fragmentation (TF) indicates the lack of continuity of the tracking system for a single target track [47].

$$TF = \frac{T_{OUT}}{NF}, \tag{23}$$

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

5.3 Comparison with state-of-the-art methods

To evaluate the tracking performance of our proposed method, we have first compared our method with the Camshift (CSH) tracker implemented in OpenCV [21] and the classical particle filter (PF) tracker [22]. To remove the camera control effect from tracking performance evaluation, CSH and PF are applied separately on the recorded experiments obtained from our fuzzy tracker (FT) method. Our assumption is that CSH and PF should perform at least as well as our tracker with the same data. All the experiments are performed first by applying the fuzzy tracker, and by recording the resulting video sequences to be tested by CSH and PF tracker methods. We have run these two trackers over the experiments recorded in Class 2 and Class 4 (see Table 3).

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 with three parameters: V_{min} and S_{min}, which are used to remove neutral colors (almost gray and almost black) in histogram computations and V_{max} to remove pixels that are too bright. We found that the best results for this tracker are obtained by using the default values for these parameters ($V_{min} = 10$, $S_{min} = 30$, $V_{max} = 256$).

We have also implemented the PF method. 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 are obtained from average frame processing rate of our method which is 6 fps for Class 2 and 4. We have obtained that 250 is the maximum number of samples that should be used to have such a frame rate. The size of these samples (width and height of the ellipses) is changed randomly by $\pm 5\%$ based on the previous target size. This value is obtained experimentally

Table 2 Tracking results of Particle Filter (PF), Camshift (CSH), and our tracker (FT) on the experiments of Class 2 and Class 4.

E	P (%)	DP (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	Tracker
	100	100	0	0.1375	0.0105	0.0336	0.0030	FT
E_6	81	82	3	0.1854	0.0791	0.0745	0.0561	PF
	72	74	10	0.3124	0.1122	0.1241	0.1015	CSH
	100	100	0	0.2068	0.0139	0.0353	0.0047	FT
E_7	83	85	5	0.1876	0.0864	0.0697	0.0571	PF
	74	75	13	0.3745	0.1127	0.1237	0.1063	CSH
	100	100	0	0.0951	0.0083	0.0329	0.0018	FT
E_8	82	84	4	0.1934	0.0925	0.0723	0.0563	PF
	75	76	8	0.3697	0.1169	0.1320	0.1072	CSH
	93	95	0.3	0.1113	0.0071	0.0353	0.0021	FT
E_9	81	82	6	0.1723	0.0967	0.0724	0.0672	PF
	73	75	12	0.3462	0.1094	0.1250	0.0967	CSH
	95	95	0.1	0.1339	0.0074	0.0398	0.0027	FT
E_{10}	82	85	7	0.1812	0.0932	0.0642	0.0552	PF
	70	72	16	0.3561	0.1148	0.1261	0.0983	CSH
	97.7	**98.08**	**0.07**	**0.1369**	**0.0094**	**0.0354**	**0.0029**	**FT**
Class 2	**81.81**	**83.62**	**4.96**	**0.1840**	**0.0896**	**0.0706**	**0.0584**	**PF**
	72.84	**74.42**	**11.75**	**0.3518**	**0.1132**	**0.1262**	**0.1020**	**CSH**
	92	93	0.35	0.1498	0.0070	0.0311	0.0021	FT
E_{16}	79	81	3	0.2035	0.0962	0.0809	0.0623	PF
	68	70	11	0.3862	0.1325	0.1401	0.1108	CSH
	100	100	0	0.1416	0.0055	0.0346	0.0018	FT
E_{17}	82	83	8	0.2067	0.1272	0.0800	0.0657	PF
	65	67	17	0.3745	0.1311	0.1325	0.1132	CSH
	100	100	0	0.1455	0.0087	0.0221	0.0018	FT
E_{18}	77	80	6	0.2125	0.0988	0.0785	0.0648	PF
	71	72	14	0.3632	0.1342	0.1368	0.1172	CSH
	91	92	0.5	0.1575	0.0165	0.0332	0.0036	FT
E_{19}	81	83	5	0.1967	0.1341	0.0794	0.0692	PF
	74	76	12	0.3741	0.1426	0.1347	0.1165	CSH
	100	100	0	0.1457	0.0122	0.0591	0.0037	FT
E_{20}	85	86	4	0.1933	0.1063	0.0806	0.0674	PF
	72	74	7	0.3815	0.1408	0.1298	0.1087	CSH
	96.47	**96.88**	**0.17**	**0.1480**	**0.0100**	**0.0360**	**0.0026**	**FT**
Class 4	**80.83**	**82.62**	**5.23**	**0.2025**	**0.1125**	**0.0799**	**0.0659**	**PF**
	69.96	**71.77**	**12.27**	**0.3759**	**0.1362**	**0.1348**	**0.1133**	**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}, PF: Particle Filter Tracker [22], CSH: CamShift Tracker [21], FT: Our tracker.

based on the maximum or minimum displacement of the target in the case where the target is approaching or moving back from the camera between two consecutive frames. For PF, the target is modeled and initialized in the same way as in our fuzzy method.

Table 2 shows the five calculated metric values for CSH and PF trackers method in comparison with our fuzzy method for experiments of Class 2 and Class 4. Our method outperforms the CSH and PF trackers. Both PF method and CSH tracker have lower target tracking precision P and more track fragmentation. The localization of the target is better with our method ($\mu_{d_{gc}}$ and $\mu_{d_{gp}}$ values are smaller). It means the location of the target is closer to the ground-truth than the CSH tracker results.

For the Camshift tracker, the results are explained by the observation made in introduction. This type of tracker relies on inter-frame proximity of the target in the image plane.

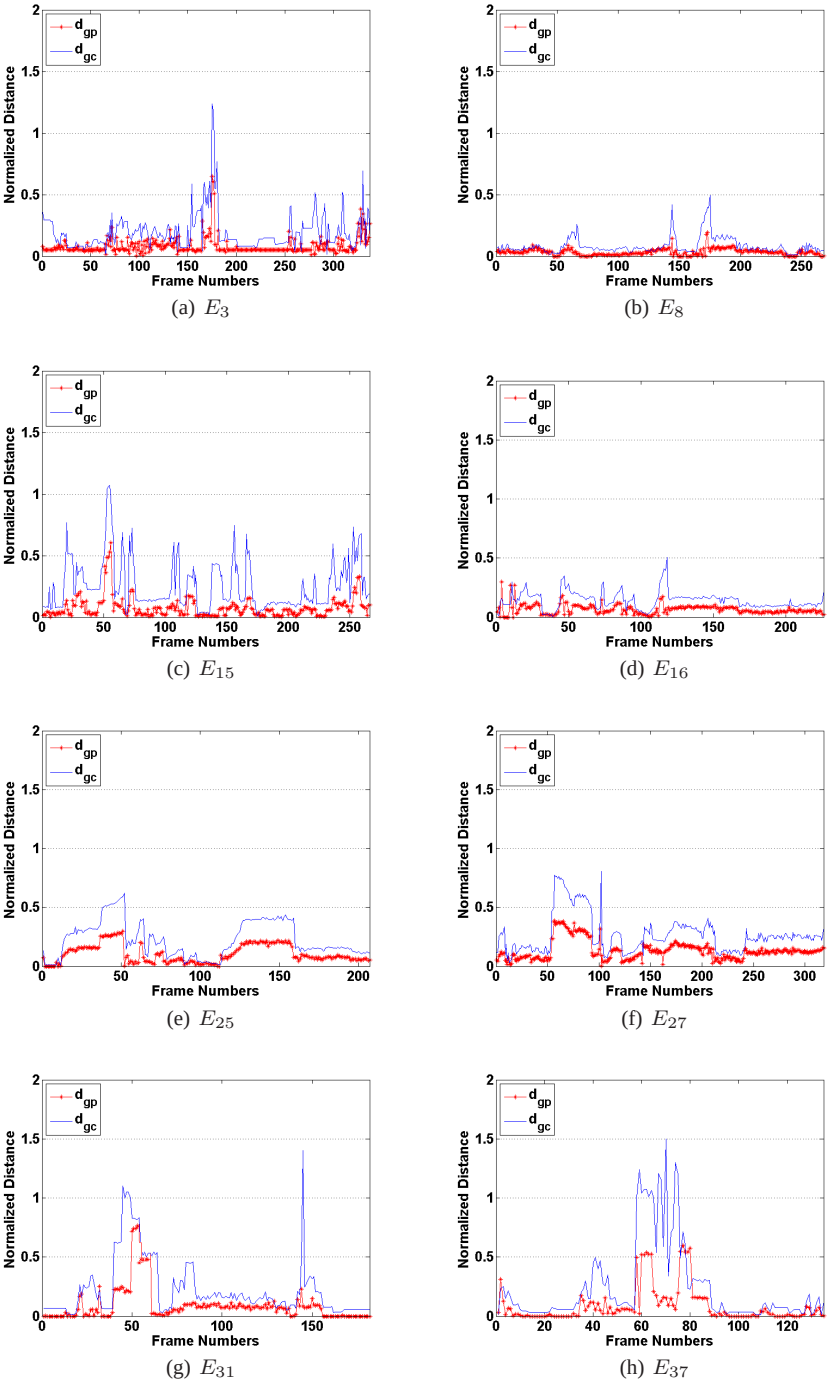
When, the camera pans or tilts, or if the frame rate is low, this assumption is no longer valid and results in poor tracking performances. For the PF, we have chosen a sampling scheme that does not rely on proximity. We just sample the image everywhere, with a larger concentration of samples in the middle of the image. This is why we get results superior to those of Camshift. However, the number of samples is not large enough to cover correctly the whole image. Thus, there are tracking failures and large localisation errors. Further, skin information are not used to position more efficiently the samples. 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 be also around areas with motion. This corresponds to the way we are sampling. This experiment shows that low frame rate and large displacements in the image plane need to be considered explicitly as proposed by our methodology.

5.4 Effect of different parameters

To evaluate the performance of our propose method for different parameters, we have done forty experiments with two IP cameras. The experiments are described in Table 3. Experiments are classified into eight classes based on different parameters such as the camera model, camera connection mode, initial model position from camera, camera control function, and image resolution. Camera connection mode may be direct or indirect. The direct mode is when the camera is connected directly to the network adaptor of the workstation. The indirect mode is when the camera is connected to the workstation through the local network. The distances of the initial model position from camera are written in Table 3. The goal is to compare the effect of different parameters such as image size, camera control function, and camera connection mode in the system performance results. Thus, in evaluation of each case, we have optimized the other parameters.

Fig. 7 shows the d_{gp} and d_{gc} values for one experiment of each class of Table 3. These distances allow us to verify if the tracker has lost the target, and if the target is near the ground-truth. If d_{gc} is more than 1, target is lost (outside of FOV). For distances smaller than 0.6, the object is in the FOV. For the range of $(0.6 < d_{gc} < 1)$, it depends if the centroid coordinates of the target are in the range [0, height-1] and [0, width-1]. For E_8, E_{16} and E_{25} the target is always in the FOV. In E_3, the target was lost one time at frame 175. For E_{27}, the target was lost one time at frame 100. For E_{15}, the target was lost two times at frames 52 and 250. In E_{31}, the target was lost several times between frames 47-55 and one time at frame 145. Finally for E_{37}, the target was lost several times between frames 58-75. For all experiments, d_{gp} is small except when the target is lost. It means that the target is well localized in most images.

Table 4 shows the results of the five metrics with the system frame rate for all experiments with image size of 640×480 and 320×240 respectively. For d_{gc} and d_{gp}, we show the mean and variance of all experiments. For classes with larger image size, the method has lost the target several times, but eventually recovers. For experiments using a smaller image resolution, there are fewer target losses (e.g. TF is smaller), smaller distance errors ($\mu_{d_{gc}}$ and $\mu_{d_{gp}}$) and the results of precision are better (P is larger). Because of d_{EP} and camera control, the error on $\mu_{d_{gc}}$ has 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 and increase $\mu_{d_{gp}}$. Indeed, the algorithm tries to find similar object to the target which is nearest to the image center. Furthermore, if the target moves very fast, to minimize d_{gp}, d_{gc} will be increased. The best results are obtained for class 2 and 4 for which the



(a) E_3 (b) E_8

(c) E_{15} (d) E_{16}

(e) E_{25} (f) E_{27}

(g) E_{31} (h) E_{37}

Fig. 7 d_{gc} and d_{gp} results for (a) E_3, (b) E_8, (c) E_{15}, (d) E_{16}, (e) E_{25}, (f) E_{27}, (g) E_{31} and (h) E_{37}

Table 3 Experiment test cases.

Classes	Experiments	Camera Model	Image Size	IMP	Connection Mode	CCF
Class 1	E_1	SNC-RZ50N	640×480	Near	Direct	RelativePanTilt
	E_2	SNC-RZ50N	640×480	Near	Direct	RelativePanTilt
	E_3	SNC-RZ50N	640×480	Far	Direct	RelativePanTilt
	E_4	SNC-RZ50N	640×480	Far	Direct	RelativePanTilt
	E_5	SNC-RZ50N	640×480	Middle	Direct	RelativePanTilt
Class 2	E_6	SNC-RZ50N	320×240	Near	Direct	RelativePanTilt
	E_7	SNC-RZ50N	320×240	Near	Direct	RelativePanTilt
	E_8	SNC-RZ50N	320×240	Far	Direct	RelativePanTilt
	E_9	SNC-RZ50N	320×240	Far	Direct	RelativePanTilt
	E_{10}	SNC-RZ50N	320×240	Middle	Direct	RelativePanTilt
Class 3	E_{11}	SNC-RZ25N	640×480	Near	Direct	RelativePanTilt
	E_{12}	SNC-RZ25N	640×480	Near	Direct	RelativePanTilt
	E_{13}	SNC-RZ25N	640×480	Far	Direct	RelativePanTilt
	E_{14}	SNC-RZ25N	640×480	Far	Direct	RelativePanTilt
	E_{15}	SNC-RZ25N	640×480	Middle	Direct	RelativePanTilt
Class 4	E_{16}	SNC-RZ25N	320×240	Near	Direct	RelativePanTilt
	E_{17}	SNC-RZ25N	320×240	Near	Direct	RelativePanTilt
	E_{18}	SNC-RZ25N	320×240	Far	Direct	RelativePanTilt
	E_{19}	SNC-RZ25N	320×240	Far	Direct	RelativePanTilt
	E_{20}	SNC-RZ25N	320×240	Middle	Direct	RelativePanTilt
Class 5	E_{21}	SNC-RZ50N	320×240	Near	Direct	AreaZoom
	E_{22}	SNC-RZ50N	320×240	Near	Direct	AreaZoom
	E_{23}	SNC-RZ50N	320×240	Far	Direct	AreaZoom
	E_{24}	SNC-RZ50N	320×240	Far	Direct	AreaZoom
	E_{25}	SNC-RZ50N	320×240	Middle	Direct	AreaZoom
Class 6	E_{26}	SNC-RZ25N	320×240	Near	Direct	AreaZoom
	E_{27}	SNC-RZ25N	320×240	Near	Direct	AreaZoom
	E_{28}	SNC-RZ25N	320×240	Far	Direct	AreaZoom
	E_{29}	SNC-RZ25N	320×240	Far	Direct	AreaZoom
	E_{30}	SNC-RZ25N	320×240	Middle	Direct	AreaZoom
Class 7	E_{31}	SNC-RZ50N	640×480	Near	Indirect	RelativePanTilt
	E_{32}	SNC-RZ50N	640×480	Near	Indirect	RelativePanTilt
	E_{33}	SNC-RZ50N	640×480	Far	Indirect	RelativePanTilt
	E_{34}	SNC-RZ50N	640×480	Far	Indirect	RelativePanTilt
	E_{35}	SNC-RZ50N	640×480	Middle	Indirect	RelativePanTilt
Class 8	E_{36}	SNC-RZ25N	640×480	Near	indirect	RelativePanTilt
	E_{37}	SNC-RZ25N	640×480	Near	Indirect	RelativePanTilt
	E_{38}	SNC-RZ25N	640×480	Far	Indirect	RelativePanTilt
	E_{39}	SNC-RZ25N	640×480	Far	Indirect	RelativePanTilt
	E_{40}	SNC-RZ25N	640×480	Middle	Indirect	RelativePanTilt

IMP: Initial model position from camera, Far: 5 meters, Near: 1 meter, Middle: 3 meters, CCF: Camera Control Function.

system frame rate is faster. TF for class 2 and 4 are the smallest. A faster system frame rate improves the results of TF, $\mu_{d_{gc}}$, $\mu_{d_{gp}}$ and P. DP values of the proposed method in all experiments is higher than the precision P, because the face may be detected correctly, but its associated region might not be selected by the fuzzy classifier. The minimum face size that can be detected by our method is 5 x 5 in smaller image resolution.

By comparing the results of class 2 and 4 with results of class 1 and 3, the effect of image resolution size for both cameras is evaluated. As shown in Table 4, with smaller resolution, faster frame rate and better tracking performance are obtained. By comparing the results of class 2 and 4 with the results of class 5 and 6 in Table 4, the effect of camera

Table 4 Experimental results of our Fuzzy tracker without zooming.

E	P (%)	DP (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	$FR(fps)$	NF	$min(\Delta x)$	$max(\Delta x)$
E_1	87	96	1.3	0.1254	0.0346	0.0533	0.0075	2.78	418	15	246
E_2	86	95	1.5	0.1576	0.0096	0.0486	0.0032	2.75	404	21	194
E_3	88	93	1.8	0.1862	0.0163	0.0503	0.0038	2.85	405	26	306
E_4	92	98	1.1	0.1457	0.0142	0.0348	0.0034	2.80	401	2	380
E_5	90	96	1.7	0.1552	0.0158	0.0438	0.0024	2.70	392	12	310
Class 1	**88.57**	**95.59**	**1.4**	**0.1540**	**0.0181**	**0.0462**	**0.0041**	**2.78**	**2020**	**11**	**380**
E_6	100	100	0	0.1375	0.0105	0.0336	0.0030	5.91	774	1	142
E_7	100	100	0	0.2068	0.0139	0.0353	0.0047	6.59	811	0	184
E_8	100	100	0	0.0951	0.0083	0.0329	0.0018	5.77	801	2	152
E_9	93	95	0.3	0.1113	0.0071	0.0353	0.0021	6.28	739	0	154
E_{10}	95	95	0.1	0.1339	0.0074	0.0398	0.0027	5.89	746	2	200
Class 2	**97.7**	**98.08**	**0.07**	**0.1369**	**0.0094**	**0.0354**	**0.0029**	**6.09**	**3871**	**0**	**200**
E_{11}	85	89	1.4	0.1578	0.0214	0.0422	0.0013	2.71	405	8	339
E_{12}	93	93	1.6	0.1674	0.0116	0.0561	0.0016	2.67	401	19	207
E_{13}	89	93	2.2	0.1600	0.0170	0.0518	0.0011	2.62	406	17	259
E_{14}	92	95	1.97	0.1385	0.0111	0.0477	0.0015	2.12	404	15	296
E_{15}	86	94	1.86	0.2346	0.0625	0.0372	0.0172	2.61	408	36	343
Class 3	**88.98**	**92.8**	**1.80**	**0.1717**	**0.0247**	**0.0470**	**0.0045**	**2.52**	**2024**	**15**	**343**
E_{16}	92	93	0.35	0.1498	0.0070	0.0311	0.0021	5.51	725	1	227
E_{17}	100	100	0	0.1416	0.0055	0.0346	0.0018	6.28	770	2	116
E_{18}	100	100	0	0.1455	0.0087	0.0221	0.0018	5.82	666	0	119
E_{19}	91	92	0.5	0.1575	0.0165	0.0332	0.0036	5.71	778	0	143
E_{20}	100	100	0	0.1457	0.0122	0.0591	0.0037	6.33	690	0	121
Class 4	**96.47**	**96.88**	**0.17**	**0.1480**	**0.0100**	**0.0360**	**0.0026**	**5.92**	**3629**	**0**	**227**
E_{21}	85	89	0.4	0.1052	0.0142	0.0436	0.0014	5.9	783	18	147
E_{22}	89	92	0.8	0.2000	0.0053	0.0891	0.0013	5.49	708	24	207
E_{23}	91	93	0.2	0.1733	0.0179	0.0681	0.0047	5.62	774	21	157
E_{24}	93	95	0.9	0.0826	0.0048	0.0324	0.0007	6.18	797	25	186
E_{25}	96	98	0.7	0.2428	0.0215	0.1060	0.0065	5.88	798	14	204
Class 5	**90.86**	**93.45**	**0.59**	**0.1608**	**0.0127**	**0.0678**	**0.0029**	**5.81**	**3860**	**14**	**207**
E_{26}	82	93	1	0.2252	0.0231	0.1047	0.0061	5	756	22	166
E_{27}	86	97	0.7	0.2690	0.0254	0.1326	0.0068	5.5	774	32	206
E_{28}	87	93	0.1	0.1397	0.0060	0.0621	0.0007	5.33	777	17	189
E_{29}	91	92	0.2	0.1658	0.0099	0.0760	0.0025	5.53	772	15	196
E_{30}	93	94	0.4	0.1712	0.0104	0.0733	0.0028	5.35	791	16	121
Class 6	**87.84**	**93.80**	**0.59**	**0.1942**	**0.0150**	**0.0897**	**0.0038**	**5.33**	**3870**	**15**	**206**
E_{31}	87	90	3.4	0.2025	0.0123	0.0672	0.0019	1.84	309	14	449
E_{32}	89	91	2.6	0.1878	0.0107	0.0578	0.0013	1.85	398	13	342
E_{33}	89	94	2.8	0.1860	0.0209	0.0474	0.0015	1.78	392	17	371
E_{34}	90	95	2.65	0.1188	0.0165	0.0265	0.0021	1.88	388	15	344
E_{35}	88	93	2.23	0.2707	0.0318	0.0601	0.0029	1.67	378	16	271
Class 7	**88.67**	**92.70**	**2.68**	**0.1932**	**0.0184**	**0.0518**	**0.0019**	**1.80**	**1865**	**13**	**449**
E_{36}	86	90	3.1	0.2773	0.0131	0.0513	0.0020	1.73	316	19	366
E_{37}	88	95	3.5	0.2866	0.0112	0.0531	0.0010	1.71	406	21	486
E_{38}	89	93	3.2	0.1624	0.0084	0.0689	0.0007	1.73	307	15	379
E_{39}	90	91	2.9	0.2194	0.0110	0.0775	0.0025	1.76	394	14	398
E_{40}	87	90	3.3	0.1819	0.0095	0.0500	0.0004	1.65	387	18	274
Class 8	**88.04**	**91.84**	**3.2**	**0.2255**	**0.0106**	**0.0602**	**0.0013**	**1.71**	**1810**	**14**	**486**

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 and NF:Number of frames, $min(\Delta x)$: minimum motion vector length, $max(\Delta x)$: maximum motion vector length.

control function for both PTZ cameras is studied. Using *RelativePanTilt* control function instead of *AreaZoom* function 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. By comparing class 1 and 3 with class 7 and 8 in Table 4, the effect of direct and indirect camera connection is studied. With direct camera connection to the computer network card, a faster system rate is obtained because of smaller network delays. By comparing the results for both cameras, results of SNC-RZ50 are better than SNC-RZ25 because of the camera characteristics such as maximum pan and tilt speed, maximum FOV, and etc. The last two columns in Table 4 and Table 2 are the minimum and maximum length of target motion vector in number of pixels. These values vary according to the image resolution size, frame rate and target movement.

Results show that our algorithm can handle and overcome large displacement (i.e. high values of $max(\Delta x)$) because of using a detect-and-match scheme and motion prediction technique that does not rely on spatial proximity. It will lose a target only if the target changes its motion direction suddenly and walks very fast in the opposite of the predicted direction (e.g. experiments with $TF \neq 0$). By using a detect-and-match scheme and combining it with a motion predictor, we can handle random motion between frames, as long as the target position is well predicted, and its appearance does not change significantly. The motion predictor is used to compensate the two delays τ_1 and τ_2 discussed in Section 3, which may cause the target to exit the FOV.

Generally, according to the mean of distances, the location of the target is near the ground-truth. The target is usually localized within 1/6th of the image diagonal from the image center. With smaller image resolution, the results of tracking are significantly better. Indeed, it results to a faster system rate because less data is transferred on the network (for the same JPEG compression level). In fact, because of larger image size, the camera sends only 16fps in 640×480, while it can send 30fps in 320×240. If the camera sends only 16fps, delay τ_1 increases. This means that a moving target will have larger motion between two frames than for 320×240. In turn, because the target have a larger displacement, the camera needs to move a larger angular distance (activate its motor longer), and thus, delay τ_2 also increases. Because delays τ_1 and τ_2 both increases (τ_2 being the more significant), larger image results into a significant performance drop. This is confirmed by comparing the 8 classes, where the mean d_{gc}, mean d_{gp}, P and DP are improved with smaller image.

In experiments with our fuzzy tracker, the target is localized with a good precision and it is about at constant distance from the image center. This can be explained by the use of motion and skin information. When localization fails, it is because of similarity or closeness of the color histogram of the target with other blobs with skin-like colors.

In all experiments, there are scale changes to verify tracking against scaling. Our algorithm can overcome scaling variations even in the image with minimum 5×5 face size (e.g. Fig.6(e) and (d)). It is because of using normalized color histogram and average color features. These two features are independent of the size of the target.

The image resolution, camera model, internet network traffic, and camera control function have effects on the system frame rate and thus on tracking error. Smaller image resolution takes less time to transfer on the network (smaller τ_1) and less processing time and results in faster frame rates. SNC-RZ50N has better performance with higher maximum pan-tilt speed changes than SNC-RZ25N. Direct camera connection removes the effect of network traffic delays from our system. Using the *RelativePanTilt* function for camera control function, the camera is always set at maximum speed which results in less tracking error and faster frame rate.

Table 5 Experiment test cases.

Class Z1	Camera Model	Image Size	IMP	Connection Mode	CCF
E_{Z1}	SNC-RZ50N	320×240	Near	Direct	RelativePanTilt + MultipleZoom
E_{Z2}	SNC-RZ50N	320×240	Near	Direct	RelativePanTilt + MultipleZoom
E_{Z3}	SNC-RZ50N	320×240	Near	Direct	RelativePanTilt + MultipleZoom
E_{Z4}	SNC-RZ50N	320×240	Near	Direct	RelativePanTilt + MultipleZoom
E_{Z5}	SNC-RZ50N	320×240	Far	Direct	RelativePanTilt + MultipleZoom
E_{Z6}	SNC-RZ50N	320×240	Far	Direct	RelativePanTilt + MultipleZoom
E_{Z7}	SNC-RZ50N	320×240	Far	Direct	RelativePanTilt + MultipleZoom
E_{Z8}	SNC-RZ50N	320×240	Far	Direct	RelativePanTilt + MultipleZoom
E_{Z9}	SNC-RZ50N	320×240	Middle	Direct	RelativePanTilt + MultipleZoom
E_{Z10}	SNC-RZ50N	320×240	Middle	Direct	RelativePanTilt + MultipleZoom

IMP: Initial model position from camera, Far: 5 meters, Near: 1 meter, Middle: 3 meters, CCF: Camera Control Function.

Our method can also recover the tracking if it loses the object (e.g. experiments with $TF \neq 0$), because of the detect-and-match scheme. Of course, it is conditional to the object being in the FOV of the camera. Occlusions are handled in the same way. However, when the object is occluded, another similar object will be tracked (the most likely candidate blob) until the occlusion ends. This could cause the real target to become out of the FOV of the camera. Fig. 6 shows an example of short-term occlusion handling. The proposed method can handle it in this case. In the reported experiments, occlusions did not cause difficulties for scene that are not crowded and people with distinctive clothing.

The random motions between frames are handled by the combination of the sampling scheme and position prediction method, as long as the target appearance does not change significantly and it remains in the camera FOV. The delays are compensated by position predictor that prevent the target from getting out of the FOV. Occlusions are handled in the same way. However, when the object is occluded, another similar object will be tracked (the most likely candidate blob) until the occlusion ends. This may cause the real target to become out of the FOV of the camera.

5.5 Tracking with camera zooming

In addition, we have studied the effect of zoom control on tracking system performance. We have done 10 experiments with SNC-RZ50 in 320×240 image resolution. The experiments are described in Table 5.

Table 6 shows the result of the five metrics, the system frame rate, and the maximum zoom level that was obtained for all experiments. We have limited the maximum zoom level of the camera (SNC-RZ50) by a factor of four to avoid target loss. However, this value was set from our farthest maximum distance, which is 6 meters. By comparing the results of class 2 and 4 of Table 4 with Table 6, there are more target losses (e.g. TF is larger), more distance errors (larger $\mu_{d_{gc}}$ and $\mu_{d_{gp}}$) and the results of precision are worst (P and DP are smaller). Camera control with both moving and zooming has lower frame rate than just moving. It is because of lens adjustment that requires additional time than only moving. The last column in Table 6 shows the maximum zoom level that was achieved in the experiment. We have been able to achieve maximum zooming only for 4 out of 10 experiments because at higher zoom level, the target is often lost and we have to zoom out to recover track and

Table 6 Experimental results of our Fuzzy tracker with zooming.

E	P (%)	DP (%)	TF (%)	$\mu_{d_{gc}}$	$\sigma^2_{d_{gc}}$	$\mu_{d_{gp}}$	$\sigma^2_{d_{gp}}$	$FR(fps)$	NF	$min(\Delta x)$	$max(\Delta x)$	m
E_{Z1}	94	96	1.3	0.2211	0.0912	0.0462	0.0140	4.62	674	2	128	3
E_{Z2}	92	93	2.1	0.2452	0.1056	0.0417	0.0162	3.59	594	6	96	3
E_{Z3}	97	97	0.7	0.1961	0.0875	0.0352	0.0137	4.41	631	3	52	4
E_{Z4}	95	96	1.2	0.2086	0.0762	0.0439	0.0155	4.83	645	5	168	4
E_{Z5}	90	92	2.6	0.2445	0.0945	0.0527	0.0149	3.06	522	1	196	3
E_{Z6}	91	93	1.9	0.2487	0.0863	0.0536	0.0176	3.69	510	7	176	4
E_{Z7}	94	95	0.95	0.2166	0.0922	0.0416	0.0144	4.11	497	4	120	3
E_{Z8}	88	90	2.62	0.2542	0.0974	0.0508	0.0150	3.93	505	6	112	3
E_{Z9}	93	93	1.1	0.2114	0.0944	0.0427	0.0146	3.49	420	10	205	3
E_{Z10}	92	92	1.2	0.2217	0.0893	0.0419	0.0155	3.35	637	2	93	4
Class Z1	**92.75**	**93.85**	**1.54**	**0.2268**	**0.0915**	**0.0450**	**0.0151**	**3.86**	**5635**	**1**	**205**	-

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 and NF:Number of frames, $min(\Delta x)$: minimum motion vector length, $max(\Delta x)$: maximum motion vector length, m: maximum zoom level.

get a wider field of view. This is explained by a low frame rate that imposes a wide field of view to keep the target inside it between two consecutive frames.

Compared to the case without zooming, a slower frame rate is obtained because of the addition of a another camera control task (zooming) that takes more time than just camera panning and tilting (zooming takes between $1.6 - 2.5$ times longer than panning and tilting). In addition, in the zooming case, lower precision, more tracking fragmentation and larger distance errors are obtained. The reasons for lower precision and more frequent tracking fragmentation are limited FOV and the delay needed for adjusting the camera zoom. Larger distance errors are because of larger target size that makes displacement seems longer when the camera is zoomed in. Furthermore, localization fails when other image areas have color histograms similar to the target. This is amplified in the case of zooming because the FOV is limited. If occlusion occurs when the camera zooms, and no other similar object is found the camera will zoom out to get larger FOV.

Fig. 8 (f) to (h), shows an example of short-term occlusion handling during zooming. The proposed method can handle it in this case. Zoom on the target is done when a good tracking is obtained (Fig. 8 (a) to (b) and (e) to (h)). The camera zooms out if the quality of tracking is not appropriate (Fig. 8 (c) to (d)).

5.6 Tracking versus available network bandwidth

We also tested our method over different network traffic loads to see how well it can track with different network delays. This is not a thorough evaluation as the traffic on the network is not a simulation of real network conditions. However, it gives an idea on how the algorithm performs when there is other activity on the network. The generated network traffic is constant and it is set to occupy a certain bandwidth. On a 100Mps network, the camera uses about 15% of the bandwidth. In our experiment, the total bandwidth used (traffic+camera) has been set to 15% (about only camera), 67% (traffic+camera) and 93% (heavy traffic+camera). In this experiment camera SNC-RZ50 is used. The camera parameters such as video compression format, level of image compression, and bandwidth control are set to the recommended values in [41].

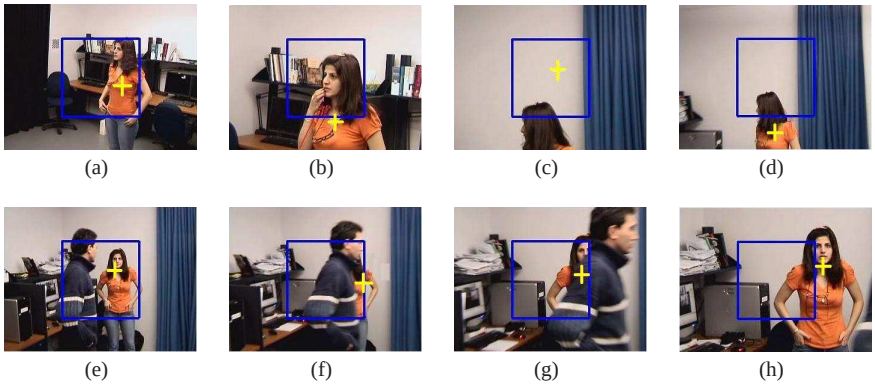


Fig. 8 Examples of tracking and zooming frames for E_3. The blue square delimits the area where we wish the target to be. (a) target tracking before zooming in, (b) target tracking after zooming in, (c) false target tracking, (d) target tracking after zooming out, (e) tracking before short term occlusion, (f) tracking while partial occlusion, (g) zooming while partial occlusion, (h) tracking and zooming after occlusion.

Table 7 Experimental results for various network bandwidth usage.

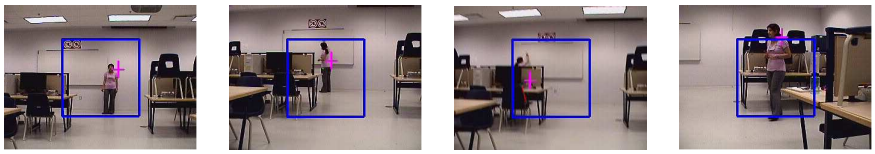
	TF (%)	$\mu_{d_{gc}}$	$\mu_{d_{gp}}$	$FR(fps)$	NF	$Used\ NetWork\ Bandwidth$ (%)
E_{L1}	2.62	0.21	0.055	1.93	403	15
E_{L2}	2.33	0.19	0.052	2.1	415	13
E_{L3}	2.48	0.18	0.05	1.97	408	14
E_{L4}	2.5	0.23	0.057	1.92	411	16
E_{L5}	2.2	0.22	0.056	1.94	409	15
$Class_{L1}$	2.44	0.20	0.054	1.97	2046	14.6
E_{L6}	3.78	0.26	0.071	1.54	361	67
E_{L7}	3.16	0.24	0.067	1.47	303	72
E_{L8}	3.08	0.25	0.66	1.62	314	71
E_{L9}	2.98	0.23	0.064	1.69	312	64
E_{L10}	2.91	0.21	0.062	1.71	306	73
$Class_{L2}$	3.26	0.23	0.06	1.59	1596	69.4
E_{L11}	7.49	0.33	0.13	1.06	237	93
E_{L12}	7.1	0.31	0.11	1.05	196	95
E_{L13}	6.8	0.36	0.09	1.08	209	96
E_{L14}	6.33	0.34	0.12	1.04	204	94
E_{L15}	6.27	0.32	0.10	1.1	215	92
$Class_{L3}$	6.88	0.33	0.11	1.06	1061	94

$\mu_{d_{gc}}$: mean of d_{gc}, $\mu_{d_{gp}}$: mean of d_{gp}, FR: System frame rate and NF:Number of frames.

Table 7 shows the effect of network utilization on the tracking performance. TF, $\mu_{d_{gc}}$ and $\mu_{d_{gp}}$ are increased as the network usage traffic is increased. It means that the traffic on the network has direct effect on the system performance as expected, because frame rate becomes slower.

6 Conclusion

In this paper, an upper body tracking algorithm for IP PTZ camera in online application is proposed. The proposed people tracking system detects, at every frame, candidate blobs



(a) (b) (c) (d)

Fig. 9 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.

using motion detection, region sampling, and color appearance. The target is detected among candidate regions using a fuzzy classifier. Then, a movement command is sent to the camera using the target position and speed.

We show that for successful tracking with a single IP PTZ camera, low frame rate and large displacements in the image plane must be account for explicitly. This is confirmed by results comparing our proposed methodology to the Camshift and particle filter method. Results show that our algorithm can handle and overcome large displacement between two consecutive frames, because it is based on combination of a detect-and-match approach and target position prediction at each frame. We will lose a target if the person changes its motion direction suddenly and walks very fast in the opposite of the predicted direction, or if areas in the scene have similar colors, including skin-like colors. We can recover the track if the target moves inside the FOV of the camera again. The proposed method can handle indirectly the short-term occlusion at the condition that the object stays in the FOV. We get better results with smaller image because the system delays are reduced. It can also detect face with small size of 5 x 5. With direct connection camera mode, smaller image resolution size and RelativePanTilt camera control function, we have better frame rate and an improvement in system performance.

The proposed method may be applied in various videosurveillance applications, including video conferencing and tracking of a speaker during a presentation. Fig. 9 shows an example of teacher tracking inside a classroom.

Future work will be enhancing robustness of the motion prediction to prevent the target from being out of the camera FOV. We will also generalize the target model and sampling scheme to be able to track a broader variety of targets (e.g. cars, buses, animals, etc.).

7 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. Ahmed, J., Jafri, M., Shah, M., Akbar, M.: Real-time edge-enhanced dynamic correlation and predictive open-loop car-following control for robust tracking. Journal of Machine Vision and Applications **19**(1), 1–25 (2008)

2. Araki, S., Matsuoka, N., Yokoya, N., Takemura, H.: Realtime tracking of multiple moving object contours in a moving camera image sequences. IEICE Transaction on Information and Sytem **E83-D**(7), 1581–1591 (2000)
3. Bagdanov, A.D., del Bimbo, A., Nunziati, W.: improving evidential quality of surveillance imagery through active face tracking. Proc. of International Conference on Pattern Recognition (ICPR) pp. 1200–1203 (2006)
4. Bellotto, N., Huosheng, H.: People tracking and identification with a mobile robot (2007). IEEE Int. Conf. on Mechatronics and Automation (ICMA)
5. Bellotto, N., Sommerlade, E., Benfold, B., Bibby, C., Reid, I., Roth, D., Fernández, C., Gool, L.V., Gonzàlez, J.: a distributed camera system for multi-resolution surveillance. Proc. of the 3rd ACM/IEEE Int. Conf. on Distributed Smart Cameras (ICDSC) (2009)
6. Bernardin, K., Camp, F., Stiefelhagen, R.: Automatic person detection and tracking using fuzzy controlled active cameras (2007). IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)
7. Boufama, B., Ali, M.: Tracking multiple people in the context of video surveillance (2007). Int. Conf. on Image Analysis and Recognition (ICIAR)
8. Bradski, G.R.: Computer vision face tracking for use in a perceptual user interface (1998)
9. Cha, S.H., Srihari, S.N.: On measuring the distance between histograms. Pattern Recognition **35**(6), 1355–1370 (2002)
10. Chan, C., Oe, S., Lin, C.: Active eye-tracking system by using quad ptz cameras. IEEE conference on industrial electronics society (IECON) **5** (2007)
11. Chen, C., Yao, Y., Jr., C., Abidi, B., Koschan, A., Abidi, M.: heterogeneous fusion of omnidirectional and ptz cameras for multiple object tracking. IEEE Trans. Circuits Syst. Video Techn **18**(8), 1052–1063 (2008)
12. Cindy, X., Collange, F., Jurie, F., Martinet, P.: Object tracking with a pan-tilt-zoom camera: application to car driving assistance. IEEE International Conference on Robotics and Automation (ICRA) **2**, 1653–1658 (2001)
13. Collins, R., Lipton, A., Kanade, T.: A system for video surveillance and monitoring. Proceedings of the American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems (1999)
14. Comaniciu, D., Ramesh, V.: Robust detection and tracking of human faces with an active camera. pp. 11–18 (2000)
15. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) **2**, 886–893 (2005)
16. Elder, J.H., Prince, S., Hou, Y., Sizintsev, M., Olevsky, E.: Pre-attentive and attentive detection of humans in wide-field scenes. International Journal of Computer Vision **72**(1), 47–66 (2007)
17. Elgammal, A., Harwood, D., Davis, L.: Non-parametric model for background subtraction. IEEE workshop on FRAME-RATE pp. 751–767 (2000)
18. Everts, I., Sebe, N., Jones, G.: Cooperative object tracking with multiple ptz cameras. International Conference on Image Analysis and Processing(ICIAP) pp. 323–330 (2007)
19. Funahashi, T., Fujiwara, T., Koshimizu, H.: Hierarchical tracking of face, facial parts and their contours with ptz camera. IEEE Int. Conf. on Industrial Technology (ICIT) **1**, 198–203 (2004)
20. Funahashi, T., Tominaga, M., Fujiwara, T., Koshimizu, H.: Hierarchical face tracking by using ptz camera. IEEE Int. Conf. on Automatic Face and Gesture Recognition (FGR) pp. 427–432 (2004)
21. Intel Corporation: Camshift tracker (2001). `http://worldlibrary.net/eBooks/Give-Away/Technical_eBooks/OpenCVReferenceManual.pdf`, [Online; accessed 21-May-2010]
22. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. International Journal of Computer Vision **29**, 5–28 (1998)
23. Kakumanu, P., Makrogiannis, S., Bourbakis, N.: A survey of skin-color modeling and detection methods. Pattern Recognition **40**(3), 1106–1122 (2007)
24. Kang, S., Abidi, B., Abidi, M.: integration of color and shape for detecting and tracking security breaches in airports. International Carnahan Conference on Security Technology pp. 289–294 (2004)
25. Kang, S., Paik, J., Koschan, A., Abidi, B., Abidi, M.: Real-time video tracking using ptz cameras (2003). 6 th Int. Conf. on Quality Control by Artificial Vision
26. Krahnstoever, N., Mendonca, P.: Bayesian autocalibration for surveillance. IEEE International conference on computer vision (ICCV) **2**, 1858–1865 (2005)
27. Krahnstoever, N., Tu, P., Sebastian, T., Perera, A., Collins, R.: 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)

28. Krahnstoever, N., Yu, T., Lim, S.: Collaborative real-time control of active cameras in large scale surveillance systems. European Conference on Computer Vision (ECCV) (2008)
29. L., Y., Payandeh, S.: Cooperative hybrid multi-camera tracking for people surveillance. Canadian Conference on Electrical and Computer Engineering (CCECE) pp. 1365–1368 (2008)
30. Leichter, I., Lindenbaum, M., Rivlin, E.: Bittracker- a bitmap tracker for visual tracking under very general conditions. IEEE T-PAMI **30**(9), 1572–1588 (2008)
31. Li, Y., Ai, H., Yamashita, T., Lao, S., Kawade, M.: Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans. IEEE T-PAMI **30**(10), 1728–1740 (2008)
32. Lim, S.N., Elgammal, A., Davis, L.: Image-based pan-tilt camera control in a multi-camera surveillance environment. Proceedings on International Conference on Multimedia and Expo (ICME) **1**, 645–648 (2003)
33. Math Forum: Points within an ellipse (2003). `http://mathforum.org/library/drmath/view/63045.html`, [Online; accessed 1-April-2009]
34. Math Open Reference: Foci of an ellipse (2008). `http://www.mathopenref.com/ellipsefoci.html`, [Online; accessed 1-April-2009]
35. Matsuyam, T., Hiura, S., Wada, T., Muease, K., Toshioka, A.: 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) **2**, 728–735 (2000)
36. Murray, D., Basu, A.: Motion tracking with an active camera. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**, 449–459 (1994)
37. Roha, M., Kima, T., Park, J., Lee, S.: Accurate object contour tracking based on boundary edge selection. Pattern Recognition **40**(3), 931–943 (2007)
38. Schreiber, D.: generalizing the lucas-kanade algorithm for histogram-based tracking. pattern recognition letters **29**(7), 852–861 (2008)
39. Shi, J., Tomasi, C.: Good features to track. IEEE conference on computer vision and pattern recognition (CVPR) pp. 593–600 (1994)
40. Sommerlade, E., Reid, I.: Information-theoretic active scene exploration. IEEE Conference on Computer Vision and Pattern Recognition pp. 1–7 (2008)
41. Sony corporation: Snc-rz25n/p cgi command manual (2005). Version 1.0
42. Tomasi, C., Kanade, T.: detection and tracking of point features. Carnegie Mellon University Technical Report **CMU-CS**, 91–132 (1991)
43. Venkatesh babu, R., Perez, P., Bouthemy, P.: Robust tracking with motion estimation and local kernel-based color modeling. image and vision computing **25**(8), 1205–1216 (2007)
44. Viola, P., Jones, J.: Robust real-time face detection. International Journal of Computer Vision **57**(2), 137–154 (2004)
45. Yao, Y., Abidi, B., Abidi, M.: 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)
46. Yilmaz, A., Li, X., Shah, M.: Contour-based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(11), 1531–1536 (2004)
47. Yin, F., Makris, D., Velastin, S.: Perforamnce evaluation of object tracking algorithms (2007). IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance(PETS)
48. Zhu, Q., Avidan, S., Yeh, M., Cheng, K.: Fast human detection using a cascade of histograms of oriented gradients. Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp. 1491–1498 (2006)