big data and cognitive computing

MDPI

Article

An Intelligent Automatic Human Detection and Tracking System Based on Weighted Resampling Particle Filtering

Liang Cheng Chang [1], Shreya Pare [2], Mahendra Singh Meena [2], Deepak Jain [3], Dong Lin Li [4], Amit Saxena [5], Mukesh Prasad [2,*] and Chin Teng Lin [2]

1 Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan; windhchs@hotmail.com
2 School of Computer Science, FEIT, University of Technology Sydney, Sydney 2007, Australia; shreya.pare@uts.edu.au (S.P.); mahendra.s.meena@student.uts.edu.au (M.S.M.); chin-teng.lin@uts.edu.au (C.T.L.)
3 Institute of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; deepak@cqupt.edu.cn
4 Department of Electrical Engineering, National Taiwan Ocean University, Keelung 202301, Taiwan; ericli@email.ntou.edu.tw
5 Department of Computer Science and Information Technology, Guru Ghasidas University, Bilaspur, Chhattisgarh 495009, India; amitsaxena65@rediffmail.com
* Correspondence: mukesh.prasad@uts.edu.au

Received: 13 July 2020; Accepted: 23 September 2020; Published: 9 October 2020

Abstract: At present, traditional visual-based surveillance systems are becoming impractical, inefficient, and time-consuming. Automation-based surveillance systems appeared to overcome these limitations. However, the automatic systems have some challenges such as occlusion and retaining images smoothly and continuously. This research proposes a weighted resampling particle filter approach for human tracking to handle these challenges. The primary functions of the proposed system are human detection, human monitoring, and camera control. We used the codebook matching algorithm to define the human region as a target and track it, and we used the practical filter algorithm to follow and extract the target information. Consequently, the obtained information was used to configure the camera control. The experiments were tested in various environments to prove the stability and performance of the proposed system based on the active camera.

Keywords: color distribution; particle filter; human tracking; codebook matching; PID controller; GMM; active camera

1. Introduction

Recently, security surveillance has applied visual-based tracking and detection techniques for improving convenience and safety for humans. Human tracking and detection are essential topics in a surveillance system. Human recognition and moving object extraction are the two parts of any typical human detection system. Human recognition identifies an object as nonhuman or human, and objects are extracted from the background by means of moving object extraction, which determines the related size and position of the object in an image. The tracking system is essentially able to predict the location during and after occlusion, as the tracked object or human is possibly occluded by other objects while tracked.

Surveillance systems typically use two kinds of the cameras: fixed camera and active camera. The fixed camera has the benefit of being low cost but comes with limited field of view (FOV), whereas

an active camera takes proper FOV as it can do pan–tilt to retain the target object within the camera scene. In addition, the latter has a better resolution since it can perform zoom in/out.

Generally, a tracking system on an active camera considers the temporal difference for extracting moving object. In this procedure, it is necessary to wait for the camera to be stable enough to process the image. In other words, the moving camera takes blurred images and extracts background pixels along with the moving object. Subsequently, the active camera operates non-smoothly and discontinuously. Hence, a particle filter tracking algorithm is applied to resolve such problem. The codebook technique is employed initially to spot the human as the target model, and after that the particle filter tracks the human by computing the Bhattacharyya distance amid the color histogram of target model with the next color histogram frame of the sampled particle position. There are various advantages of using a color histogram such as efficient computation, tracking of nonrigid objects, robustness to partial occlusion, scale invariant, and rotation.

In this paper, a real-time human tracking system is constructed with an active camera and has the following characteristics:

- Rapidly detects a human
- Tracks an object by not considering background information
- Handles occlusion conditions
- Operates an active camera continuously and smoothly
- Appropriately zoom in/out

2. Related Work

There are four key parts in our entire system: image source, human detection, human tracking, and camera control, as described in Figure 1. As a quick review of our procedure, we set the initial FOV as the scene we wanted to capture. Then, we detect and extract an object recognized as a human. We track the human object and use its moving information to pan–tilt–zoom (PTZ) the camera via a proportional-integral-derivative (PID) controller so that the target stays in the center of the FOV.

A human detection system finds the position and size of the human in an image. Optical flow [1,2] is considered in order to estimate a moving object independently at the cost of complex computations. Zhao and Thorpe [3] proposed a stereo-based segmentation technique for extracting objects from the background and then recognize the objects using neural network. While techniques based on stereo vision are more robust, it needs a minimum of two cameras, and it fails to perform well in long-distance detection. Viola et al. [4] proposed a cascade architecture detector, where adaptive boosting (AdaBoost) iteratively builds a robust classifier guided by performance criteria that are specified by user. The cascade method swiftly rejects non-pedestrian samples in the early cascade layer; thus, processing speed of this approach is high. The templates in a template-based approach [5] have short sequences of 2D silhouettes gained from motion capture data. This method detects human silhouettes having a particular walking pose. To rapidly spot humans, a shape-based human model is chosen, and codebook matching is used to classify a human. This reduces the time taken in detecting humans from the other objects. Montabone and Soto [6] proposed a novel computer vision technique that can operate moving cameras and spot a human in various poses in the case of a complete or partial appearance of the human. Pang et al. [7] presented an efficient histogram of a gradient-based human detection technique. A human tracking system follows a human target through the sequence of images regarding changes in scale and position. Between the several tracking methods, we analyzed three to synthesize our research.

First, feature-based tracking, a very common method, tracks features by motion, edge, or color using edge detecting methods such as the Sobel approach, Laplacian approach, and Marr–Hildreth approach [8,9]. These techniques use masks to perform convolution over an image for edge detection. Li et al. [10] proposed a 3D human motion tracking system with a coordinated mixture of factor analyzers. Lopes et al. [11] designed a hierarchical fuzzy logic-based approach for object tracking.

It uses a complicated and large set of rules, has a long computation time, and the pixels at the edges are not always continuously detected. The abovementioned approach uses gray scale images for edge detection, and we chose not to use this for color images because of information loss on the color space vector. Moreover, edge detection in a gray scale image cannot be robust and sufficient.

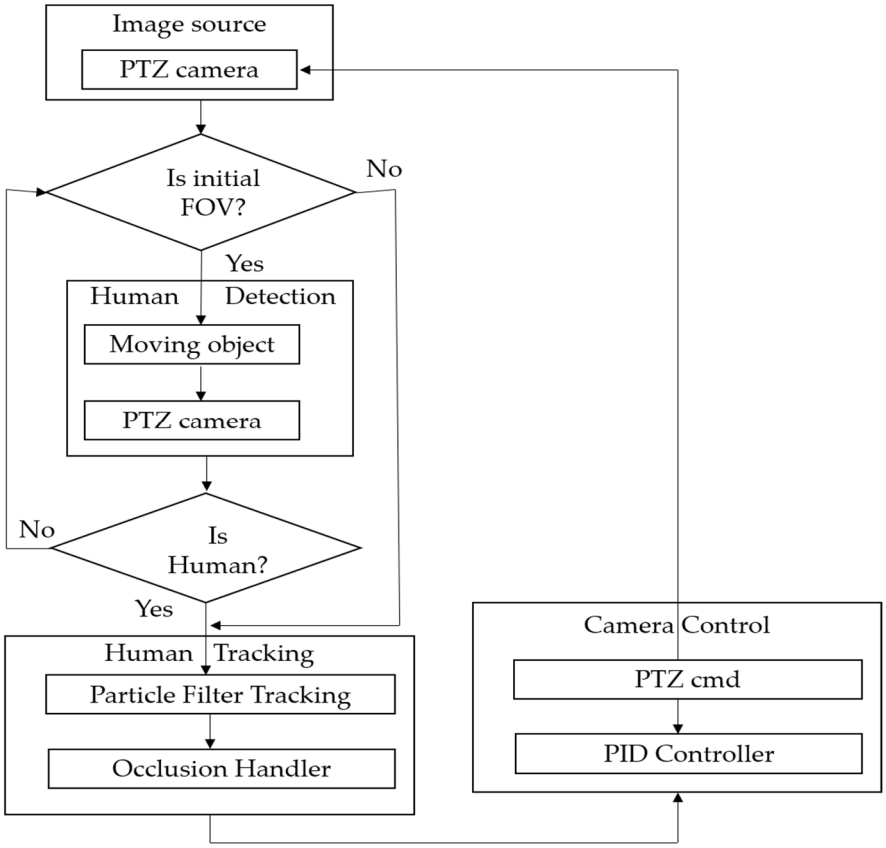


Figure 1. Overview of the system.

Secondly, pattern recognition methods learn the objects at the target and find it in sequential images. Williams et al. [12] extended the method to a relevance vector machine (RVM) that learns a nonlinear translation predictor. Collins et al. [13] proposed a mechanism for an online feature selection mechanism that can be used for multiple features evaluation. The presented approach tracks and adjusts the features set for improving tracking performance. The feature evaluation mechanism is embedded in a mean-shift tracking system. It can adaptively select tracking features. Zhang et al. [14] proposed a robust 3D human pose tracking approach from silhouettes using a likelihood function. Zhao et al. [15] used a principal component analysis to extract features from color and use them in a random walker segmentation algorithm to assist human tracking.

Thirdly, there are gradient recognition methods with a focus on pattern recognition, such as the mean-shift algorithm. Fukunaga and Hostetler [16] initially proposed the mean-shift algorithm for clustering data. Comaniciu et al. [17] proposed a kernel-based object tracking method, where object region tracking is denoted using a spatially weighted intensity histogram, and its similarity rate is computed using Bhattacharyya distance following an iterative mean-shift technique. Many applications [18–21] later proposed various mean-shift algorithm variants. Even though the mean-shift object tracking technique is well-performed over sequences with comparatively slight object displacement, its performance cannot be guaranteed in the case where objects suffers full or partial occlusions. Kalman filter [22,23] and particle filter [24,25] algorithms are considered along mean-shift algorithms for improving the tracking performance under partial occlusion. The approach by Bhat et al. [24] uses a fusion of color and KAZE features [26] in the particle filter framework to

give an effective result in different environments for tracking the target. Still, this approach requires a strategy for fast failure occlusion recovery for the post-occlusion target recovery. To track multiple targets by deploying the same color description with cancelation functionality and internal initialization, Nummiaro et al. [25] proposed a color particle filer embedded along a detection algorithm. Our major contribution in this work is a novel multitarget tracking algorithm that incorporates particle filters with a Gaussian mixture model to improve tracking accuracy and computational efficiency. In order to detect humans fast, we chose the shape-based human model to classify humans by codebook matching, which decreases the time of human detection compared to the other objects.

Many tracking systems work only on PTZ because to keep the object in FOV, an active camera can be pan–tilt and can utilize zoom in/out for adjusting resolution, thus keeping the tracked object with a well-proportioned resolution regarded to the FOV. Morphological filtering of motion images were used by Murray et al. [27] to perform background compensation. Using an active camera mounted on a pan/tilt platform, Murray's technique can successfully track a moving object from dynamic images. A kernel-based tracking method was used in the proposed system to overcome the apparent background motion on a moving camera. Karamiani and Farajzadeh [28] considered feature points' information of direction and magnitude to detect camera motion accurately. The method is used for detecting multiple moving object accurately in active and fixed camera models. Lisanti et al. [29] proposed a method that enables real-time target tracking in world coordinates, and the method offers continuous adaptive calibration of a PTZ camera. Mathivanan and Palaniswamy [30] used optimal feature points and fuzzy feature matching to accomplish human tracking. In the context of the tracking applications of humans using deep learning, Fan et al. [31] proposed human tracking and detection using a convolutional neural network for partial occlusion and view, scale, and illumination changes. Tyan and Kim [32] proposed a compact convolutional neural network (CNN) based visual tracker in conjunction with a particle filter architecture. A face tracking framework based on convolutional neural networks and Kalman filter was proposed for the real-time detecting and tracking of the human face [33,34]. Luo et al. [35] proposed a matching Siamese network and CNN-based method to track pedestrian. The method used a faster-R-CNN to distinguish pedestrians from surveillance videos. However, the method still requires target occlusion to be resolved in order for it to be a more robust real-time pedestrian tracking tool. Xia et al. [36] proposed method tracks single and multi-objects in long-term tracking in real time, which determine and identify the target bounding box in a traffic scene, CNN is firstly trained. Then, a particle filter (PF) is used as the tracker to implement the preliminary multi-object tracking. A particle filter and neural network learning evaluated in person re-identification scenario was proposed in [37], while a hybrid Kalman particle filter (KPF) for human tracking was proposed in [38]. KPF is more time-consuming, especially in the case of non-occlusion. Real-time performance of the proposed filter is not good in terms of speed.

The deep learning models are time inefficient and costly in terms of memory as they tend to expand large number of nodes, which results in large computation. Such models mostly fail in real-time applications, and their implementation requires high-end processors. Therefore, complexities of the network need to be reduced to decrease the computation time and limit the number of computations [37]. The advantage of the proposed method is its simplicity and ease of implementation. The proposed models can be executed on a simple CPU for the real-time videos. Thus, it is an efficient approach as well.

In this research, we used a wide-angle camera to find the target, and then camera calibration methods gave the active camera pan–tilt commands to keep the target in the center of the FOV and for specific object position tracking. In the case where the size of the target was larger or smaller than a maximum or minimum predefined size, then the zoom in/out command was used accordingly.

3. Proposed System

This section describes each algorithm and method used in this paper. Figure 2 shows the three categories of the tracking system. To detect a human, we first extracted moving objects from

the image source and then used codebook matching for each one of them to be categorized as human and non-human.

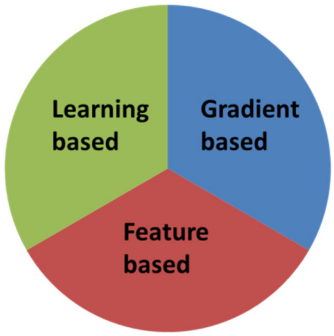


Figure 2. Three categories of tracking systems.

3.1. Human Detection

In the majority of the surveillance systems, the position of the camera is fixed, whether it is a static camera or active camera. The fixed position of the camera allows for extraction of a moving object by using background subtraction. To make the method computationally efficient, background subtraction uses only gray level images. This will also make our system more efficient when using it in real time situations. The first image frame can be adjusted over time using Equation (1), which is used to construct the background, where I_B^{n-1} and I_B^n represent previous and current background images, respectively.

$$I_B^n(x,y) = \begin{cases} \alpha * I_B^{n-1}(x,y) + (1-\alpha) * I_c(x,y), & I_M(x,y) = 0 \\ I_B^{n-1}(x,y), & I_M(x,y) = 1 \end{cases} \quad (1)$$

Scaling factor $\alpha(0,1)$ was used to update the background image. Active pixels between frames n and $n-1$ are represented by $I_M(x,y)$.

To determine the moving object, the current image I_c is subtracted from the background image I_B as described in Equation (2). To obtain the binary moving object M_{obj}, threshold ths is applied to results of Equation (2) using Equation (3).

$$I_{BS}(x,y) = \left| I_c(x,y) - I_B(x,y) \right| \quad (2)$$

$$M_{obj}(x,y) = \begin{cases} 1, & I_{BS} \geq ths \\ 0, & I_{BS} < ths \end{cases} \quad (3)$$

The details of the moving object and codebook matching are indicated in Figure 3. The binary threshold image M_{obj} undergoes a dilation process to fill holes of moving objects and to enlarge the boundaries. The step by step process is shown in Figure 4.

Human-shape information was used to build our codebook matching algorithm. The extracted moving object was normalized into a 20×40 pixels image. The position of the shape pixels in the image was extracted by the shape feature extraction. These features are pointed by red dots in Figure 5; 10 Y-axis coordinates are chosen from the object's rightmost and leftmost boundary, and 20 coordinates of the corresponding X-axis are arranged as a feature vector. The vectors are shown by blue blocks in Figure 5. As shown in Figure 5, there are a total of 10 bins in the histogram, represented by green blocks. As a result, there are 30 features vectors representing a human object.

We can conclude by observation that the top and bottom shape pixels of the Y-axis cannot be chosen as feature points as these pixels are changeable. The method used to select Y-axis coordinates is to firstly calculate the standard deviation of the reach value of Y-axis in the training sample, and then select the 10 lowest standard deviation values from each side.

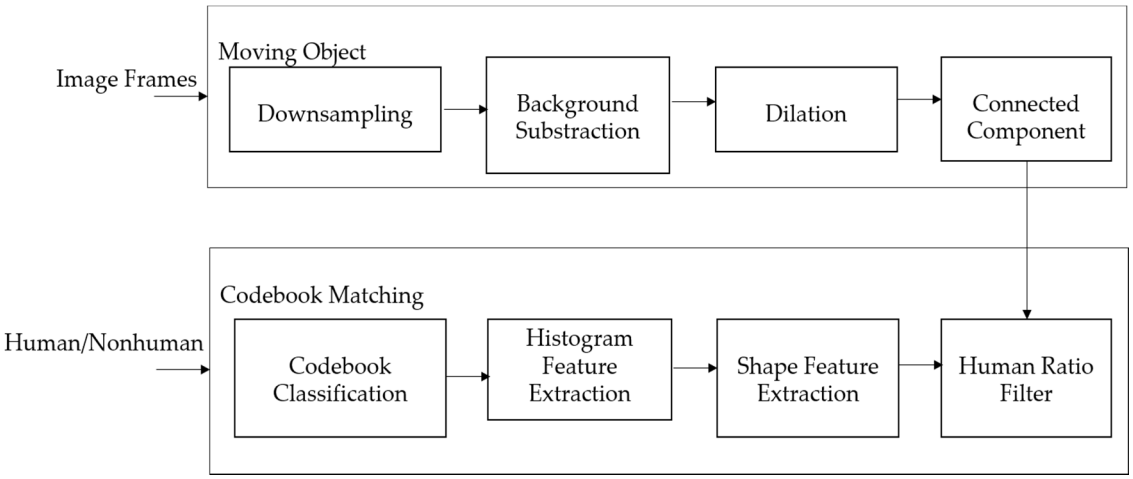


Figure 3. Human detection system.

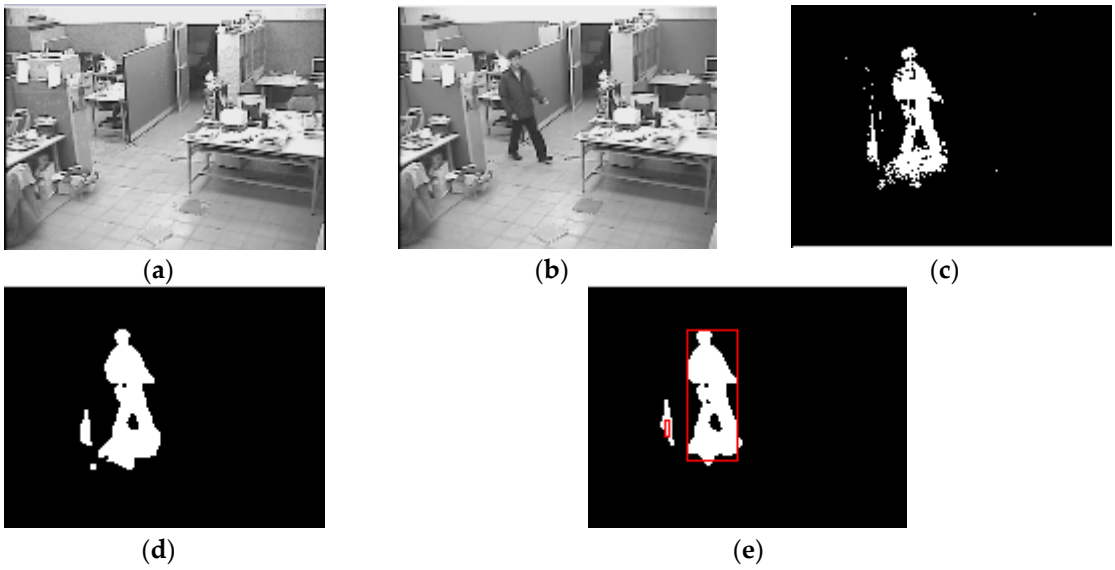


Figure 4. Step by step process of moving object extraction. (**a**) Background image I_B; (**b**) current image I_c; (**c**) binary moving object M_{obj}; (**d**) dilated image I_D; (**e**) region of interest I_{ROI}.

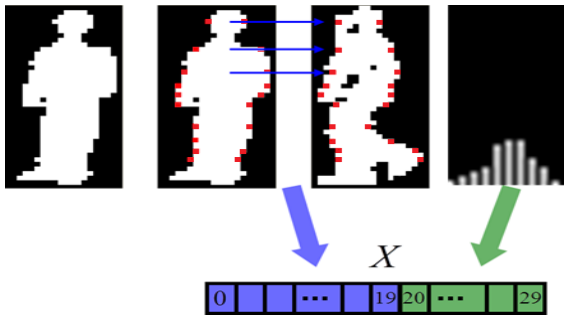


Figure 5. Example of a feature vector X.

A list of feature vectors was represented by the codebook. Matching of the feature vector and codebook vectors was done to find the minimum distortion code vector in comparison to the object feature vector. We can say that X denotes a series of feature vectors including M-dimensional data, designated by $X^0 \dots X^i \dots X^{(M-1)}$. Code words V are defined as $V_0 \dots V_j \dots V_{(N-1)}$ and have N sets

each in codebook C. Similar to the feature vector, each code word has M-dimensional data defined as $V_j^0 \dots V_j^i \dots V_j^{(M-1)}$. Distortion between code words and feature vectors was defined by Equation (4).

$$Dis_j = \|X - V_j\| = \sum_{i=0}^{M-1} \left| X^i - V_j^i \right| \tag{4}$$

$$Dis_{min} = min\left(Dis_j\right) \;\; j = 0 \dots N-1 \tag{5}$$

If the value of Dis_{min} in Equation (5) is less than the threshold, it is assumed that feature vector X and the moving object it represented was of a human, and if the value of Dis_{min} is greater than the threshold we then assume that it is a nonhuman object. The demonstration of comparing X with V_j is shown in Figure 6.

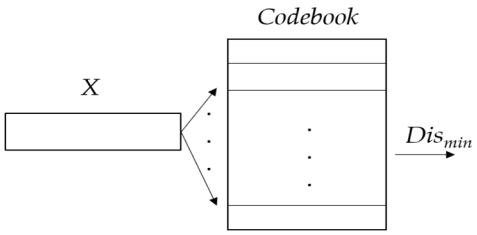


Figure 6. The procedure of the comparison with the codebook.

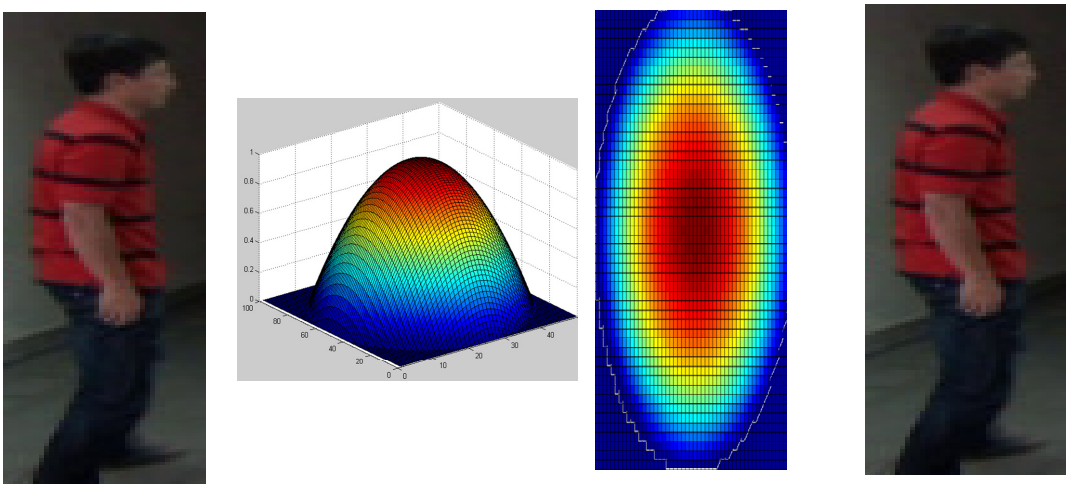
3.2. Human Tracking

A particle filter algorithm was proposed in the study, which is based on a weighted resampling particles method. In this algorithm, high weighted samples were selected for the human tracking system. The basic idea of our particle filter is to approximate the probability distribution by weighted sample sets. One hypothetical state of the object with corresponding discrete sampling probability is represented by each sample [25].

Colored information is more accurate compared to grayscale information if we use color as the feature for the purpose of object tracking. For our experimentation we chose HSV (Hue, Saturation, and Value) color space for better performance of tracking compared to RGB (Red, Green, Blue) color space because of its ability to reduce lightness and illumination sensitivity. Every color channel was represented by 8 bits, which in turn produces $256 \times 256 \times 256$ bins of the color histogram. Color data are quantized into $6 \times 6 \times 6$ without generality loss, thus making the entire bin of color histogram as 216 bins. To represent the target object, kernel function was used. The Epanechnikov kernel function was selected to represent the target object to introduce a spatially-smooth function to reduce the search on small neighborhood region. The convex and monotonically decreasing Epanechnikov kernel was selected to mask the target's density estimate spatially. The rationale of using the kernel as a weighted mask is to assign smaller weights to the pixels farther away from the center of the target, since those pixels are often affected by occlusion or interference from the background. Figure 7b shows the Epanechnikov kernel. This kernel function has the highest value at the center of distribution. If we look at the Region of Interest (ROI) of the target model in Figure 7a, the pixels that are closer to the center of the ROI contain more important information, and the background pixels are mostly near the ROI's boundary. The Epanechnikov kernel function was selected to represent the target object as it is computationally simple and can disregard the boundary information. This kernel performs well in terms of improved stability, accuracy, and robustness on camera motion and partial occlusions. Epanechnikov kernel is defined by Equation (6), where x represents normalized pixels in the region

defined as the target model. When the proposed kernel function is applied to the target model, more critical information is contained by pixels closer to the ROI center, as shown in Figure 7.

$$k(x) = \begin{cases} \frac{3}{4}(1-x^2) & if \quad \|x\| \leq 1 \\ 0 & otherwise \end{cases} \tag{6}$$



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

Figure 7. (**a**) Target object. (**b**) Epanechnikov kernel function. (**c**) Kernel function. (**d**) Target object and kernel function.

A robust tracking framework is provided by the particle filter algorithm, as it represents uncertainty. The algorithm is capable of keeping its options open and at same time it is also capable of considering multiple state hypotheses. Temporary occlusions can be dealt with by the particle filter as less likely object states will be part of the tracking process temporarily [25]. Occlusion handler steps and weighted resampling are the two basic differences between the original tracking method and our tracking method. Our proposed tracking method is shown in Figure 8. The differences between the original particle filter and ours are weighted resampling and occlusion handler.

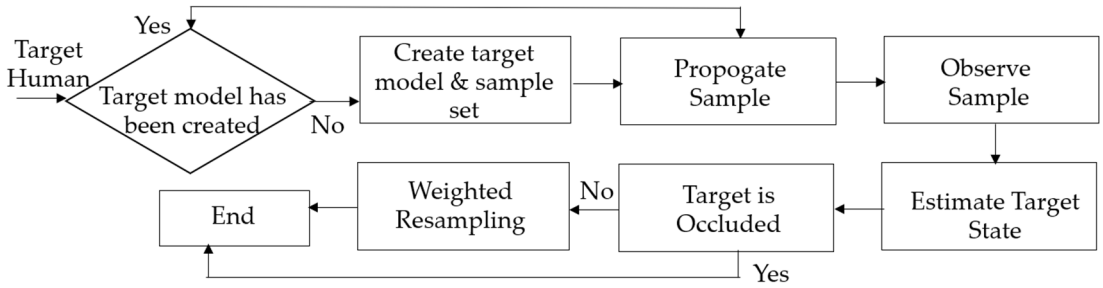


Figure 8. Step-by-step process of the weighted resampling particle filter.

The first step in the process of weighted resampling particle filter is to define the target model. It can be defined in Equation (7) at location y as m-bin histogram $q_y = \left\{q_y^{(u)}\right\}_{u=1\ldots m}$. The normalization factor f can be represented by Equation (8); δ is the Kronecker delta function, while I is the number of pixels in the ROI region, and $a = \sqrt{w^2 + h^2}$ is used as the normalization factor for the size of the object region.

$$q_y^{(u)} = f \sum_{i=1}^{I} k\left(\frac{\|y - x_i\|}{a}\right) \delta[h(x_i) - u] \tag{7}$$

$$f = \frac{1}{\sum_{i=1}^{I} k\left(\frac{\|y-x_i\|}{a}\right)} \tag{8}$$

The sample model $p_y = \left\{p_y^{(u)}\right\}_{u=1\ldots m}$ is represented in the same way as the target model. Bhattacharyya distance d is used to measure the distance between the sample and target model; it is termed as similarity value ρ. If the value is large, the two models are considered similar, whereas if the value of ρ is equal to 1, it implies that the histogram of the sample and the target model is identical.

$$p_y^{(u)} = f\sum_{i=1}^{I} k\left(\frac{\|y-x_i\|}{a}\right)\delta[h(x_i)-u] \tag{9}$$

$$\rho[p,q] = \sum_{u=1}^{m}\sqrt{p^{(u)}q^{(u)}} \tag{10}$$

$$d = \sqrt{1-\rho[p,q]} \tag{11}$$

In the particle filter algorithm, the target model can also be represented by state vector *s_target*. It is defined in Equation (12) where w and h represent the width and height of ROI, respectively; $(x,\ y)$ represents the center of ROI, and $\left(v_x, v_y\right)$ represents the motion of the object. Equation (13) is used to compute the initial sample set $S_{initial} = \left\{s^{(n)}\right\}_{n=1\ldots N}$ where I is an identity matrix, *r.v.* is a multivariate Gaussian random variable, and N represents the number of samples. A dynamic model is represented by Equation (14), which propagates the sample; the deterministic component of the model is represented by A. The target human size and position can be determined from the estimated vector using the weight of every sample and its state vector, as shown in Equation (15). To update the weight of each sample, Bhattacharyya distance is used and is shown in Equation (16).

$$s_{target} = \left\{\mathrm{x}, v_x, \mathrm{y}, v_y, \mathrm{w}, \mathrm{h}\right\} \tag{12}$$

$$s^{(n)} = Is_{target} + r.v. \tag{13}$$

$$s_t = As_{t-1} + r.v._{t-1} \tag{14}$$

$$E[S_t] = \sum_{n=1}^{N}\omega_t^{(n)}s_t^{(n)} \tag{15}$$

$$\omega^{(n)} = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{d^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(1-\rho[p_{s^{(n)}},q])}{2\sigma^2}} \tag{16}$$

The resampling step in the process of the weighted resampling particle filter is used to avoid the degeneracy of the algorithm, which means, it prevents the situation where most of the sample weights are close to zero. To determine the need and time of resampling step, Equations (17) to (19) can be used; in $rate \in (0,1)$, N_{ths} and N_{eff} represent the given threshold sample and the effective number of samples, respectively.

$$N_{eff} < N_{ths} \tag{17}$$

$$N_{eff} = \frac{1}{\sum_{n=1}^{N}\left(\omega_t^{(n)}\right)^2} \tag{18}$$

$$N_{ths} = rate * N \tag{19}$$

In the process of resampling, sample selection depends on weights; high weight samples may be selected a number of times, which will lead to a number of copies of those samples, and relatively low weight samples may not get selected at all. Given a sample set S_{t-1} and the target model q, for the first

iteration, S_{t-1} is set to $S_{initial}$. The details of the particle filter algorithm for each iteration is described as follows:

1. Propagate each sample from the set S_{t-1} by a linear stochastic differential equation:

$$s_t^{(n)} = As_{t-1}^{(n)} + r.v._{t-1}^{(n)}$$

2. Observe the color distributions:

   (a) Calculate the color distribution: $p_{s_t^{(n)}}^{(u)} = f\sum_{i=1}^{I} k\left(\frac{\|s_t^{(n)} - x_i\|}{a}\right)\delta[h(x_i) - u]$ for each sample in the set S_t

   (b) Calculate the Bhattacharyya coefficient for each sample of the set S_t: $\rho\left[p_{s_t^{(n)}}, q\right] = \sum_{u=1}^{m}\sqrt{p_{s_t^{(n)}}^{(u)} q^{(u)}}$

   (c) Weight each sample of the set S_t:

$$\omega_t^{(n)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(1-\rho[p_{s_t}(n),q])}{2\sigma^2}}$$

3. Estimate the mean state of the set S_t: $E[S_t] = \sum_{n=1}^{N} \omega_t^{(n)} s_t^{(n)}$
4. Resample the sample set S_t, if $N_{eff} < N_{ths}$: Select N samples from the set S_t with probability $\omega_t^{(n)}$:

   (a) Calculate the normalized cumulative probabilities c_t': $c_t^{(0)} = 0$; $c_t^{(n)} = c_t^{(n-1)} + \omega_t^{(n)}$; ${c'}_t^{(n)} = \frac{c_t^{(n)}}{c_t^{(N)}}$

   (b) Generate a uniformly distributed random number $r \in [0, 1]$.

   (c) Use binary search to find the smallest j for which ${c'}_t^{(j)} \geq r$.

   (d) Set ${s'}_t^{(n)} = s_t^{(j)}$.

   Finally, resample by $S_t = S_t'$.

In the initial resample step of the particle filter, samples were selected randomly, so it is possible that the selected sample has a relatively low weight, and the process ended up tracking different objects and considering them as target object, which decreased tracking accuracy, as shown in Figure 9. Figure 9 shows the sample points with high weights are in the ROI (green block), and samples with relatively low weights are in the red block. Although two blocks have nearly the same similarity value, the actual target object is in the green block. Consequently, it may track a different object as the target object. In other words, it will decrease the accuracy of tracking. Thus, we proposed a weighted resampling algorithm to cover this problem. The proposed algorithm of weighted resampling prevents this problem. First, the top sample is selected and set to S_t^{top} with N_{top} weights from set S_t, as shown in Equations (20) to (21). The parameter *top* represents the top rate and for our experiment it is set to 0.2. The S_t^{top} only selects samples with the top 20% weights from set S_t.

$$N_{top} = top * N \tag{20}$$

$$S_t^{top} = \left\{s^{top(n)}\right\}_{n=1...N_{top}} \tag{21}$$

N samples were reproduced in S_t according to the weight of $s^{top(n)}$. This step will produce $s^{top(n)}$, which has a relatively larger number of times in S_t, and others with relatively low weight will be produced at least once. Figure 10 shows the samples points with high weights are in the ROI (green

block), and samples with relatively low weights are in the red block. Figure 11 shows the weighted resampling result. Most of sample points lie in the green block or in the target object region. A Gaussian mixture model (GMM) was applied to update the target model over time. For approximation of any continuous probability distribution K, Gaussian distributions have been used. The GMM [39] is a robust method for dynamic backgrounds. It is mostly used due to its robustness to various background variations like multi-modal, quasi periodic and gradual illumination changes. GMM is a semiparametric multimodal density model consisting of a number of components to compactly represent pixels of image block in color space with illumination changes. Therefore, a Gaussian mixture model (GMM) was applied to update the target model over time. The image can be represented as a set of homogeneous regions modeled by a mixture of Gaussian distributions in color feature space. In comparison, non-Gaussian mixture models [40] present an image without taking spatial factor into computation. Gaussian distribution $N(x|\mu_k, \sigma_k)$ with mean μ_k and standard deviation σ_k was considered here. The weight of Gaussian distribution is represented by π_k, and sum of all weights is equal to 1. Equation (22) describes the process of Gaussian mixture model (GMM).

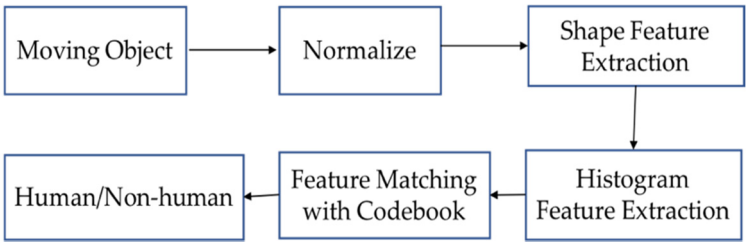


Figure 9. Flow diagram for codebook matching.

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \sigma_k) \tag{22}$$

Figure 10. Original resampling algorithm result. Samples for the actual target are in the green area, i.e., our ROI, while the red area has samples with relatively low weights.

Figure 11. Weighted resample, with most of the sample points lying in the green area, our real ROI.

The GMM update algorithm is applied to update the color histogram of the target model; $K = 3$ Gaussian distributions is used to model each bin $q^{(u)}$. The mean μ_k, standard deviation σ_k, and weight π_k were initialized respectively as $\mu_k = q^{(u)}$, $\sigma_k = 1$, and $\pi_k = \frac{1}{K}$, where $k = 1 \sim K$.

1. We sorted $\{\pi_k\}_{k=1\sim K}$ in descending order and obtained the order $\{\pi_a, \pi_b, \pi_c\}$, $\pi_a \geq \pi_b \geq \pi_c$.
2. We updated the bin's value using Equation (23) where $A = 0.6$, $B = 0.25$, $C = 0.15$, and a, b, c was the descending order.

$$q'^{(u)} = A\mu_a + B\mu_b + C\mu_c \tag{23}$$

3. If the difference between the previous and current frames' $q^{(u)}$ was smaller than the threshold, we used Equation (24) to find the first Gaussian distribution where k follows the descending order $\{ a, b, c \}$.

$$\left|q^{(u)} - \mu_k\right| < \sigma_k * 3 \tag{24}$$

If we successfully find the Gaussian distribution by Equation (24), it would update μ_k, σ_k, π_k by Equations (25) to (27), where $\alpha = 0.05$ and $\beta = 0.01$, and the other weights would be updated by $\pi_j = (1 - \beta) * \pi_j$ where $j = 1 \sim K$ and $j \neq k$.

$$\mu_k = (1 - \alpha) * \mu_k + \alpha * q^{(u)} \tag{25}$$

$$\sigma_k = \sqrt{(1 - \alpha) * \sigma_k * \sigma_k + \alpha * \left(q^{(u)} - \mu_k\right)^2} \tag{26}$$

$$\pi_k = (1 - \beta) * \pi_k + \beta \tag{27}$$

These steps produced the updated target model $q' = \left\{q'^{(u)}\right\}_{u=1\ldots m}$. The proposed occlusion handler was color-based. The algorithm equated similarities between the target model and candidate model. Figure 12 shows the flowchart of the occlusion handler. The following is the step-by-step process of the proposed occlusion handler:

1. Candidate model $c = \left\{c^{(u)}\right\}_{u=1\ldots m}$ ROI was created in the current frame.
2. The similarity value between target model $q' = \left\{q'^{(u)}\right\}_{u=1\ldots m}$ and candidate model $c = \left\{c^{(u)}\right\}_{u=1\ldots m}$ was computed.
3. If similarity was less than ths_{sim}, resampling was not performed, and it was assumed that the candidate model was occluded by another object.
4. The count was increased using $Count = Count + 1$.
5. Step 1–4 were repeated during the tracking process to see whether the similarity value becomes larger than ths_{sim}, the tracked human appeared or $Count \geq 10$. Termination condition avoids the spreading of the samples out of the image. Figure 13 shows the images for frame T, T+4, T+9, T+14 using proposed occlusion handler.

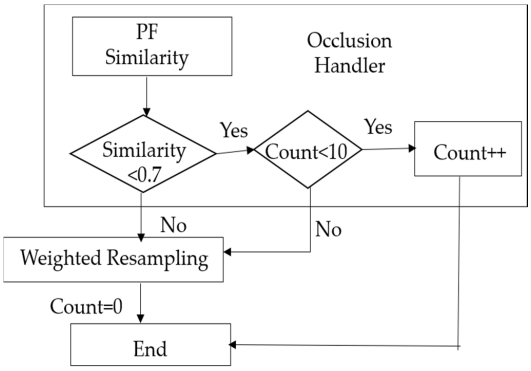


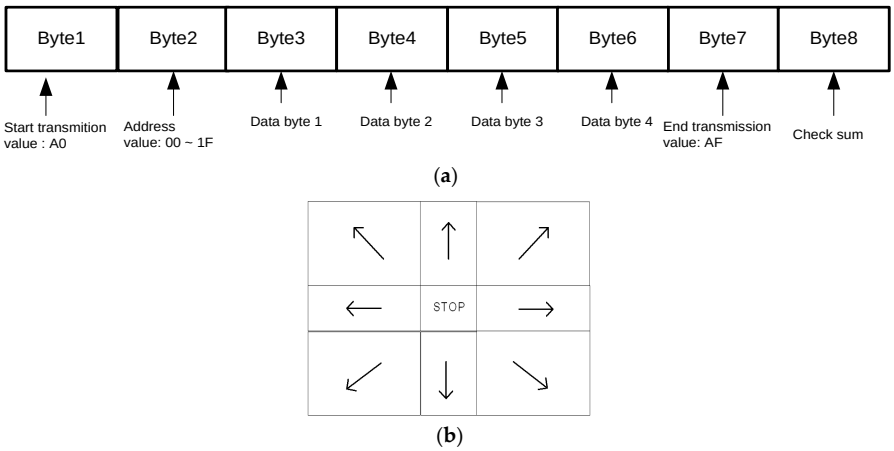
Figure 12. Occlusion handler flow chart.

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

Figure 13. Proposed occlusion handler: (**a**) Frame T; (**b**) Frame T+4; (**c**) Frame T+9; (**d**) Frame T+14.

3.3. Camera Control

Pelco P-protocol [31] was used to control the active camera through an RS-232 to RS-485 converter. The protocol allows us to have control over pan, zoom step, and tilt angle to achieve effective tracking. The active camera is controlled by pelco P-protocol [34] through the RS-232 to RS-485 converter. It needs to control pan (horizontal direction), tilt (vertical direction) angle, and the zoom's step to achieve tracking purpose. The pelco P-protocol has 8 bytes data with message format as shown in Figure 14a. Byte 1 and Byte 7 are the start and stop bytes, respectively, and they are always set to 0xA0 for Byte 1 and 0xAF for Byte 7. Byte 2 is the receiver or camera address. In this thesis, we only used one camera, so Byte 2 is always set to 0×00. Byte 3, Byte 4, Byte 5, and Byte 6 are used to control the pan–tilt–zoom (PTZ) as shown in Table 1. The last byte is an XOR check sum byte.



(**a**)

(**b**)

Figure 14. (**a**) Message format. (**b**) Field of view (FOV) divided into 9 regions associated with control directions.

Table 1. Data byte 1 to 4 format.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data byte 1	Fixed to 0	Camera On	Auto Scan On	Camera On/Off	Iris Close	Iris Open	Focus Near	Focus Far
Data byte 2	Fixed to 0	Zoom Wide	Zoom Tele	Tilt Down	Tilt Up	Pan Left	Pan Right	0 (for pan/tilt)
Data byte 3	Pan speed 00 (stop) to 3F (high speed) and 40 for Turbo							
Data byte 4	Tilt speed 00 (stop) to 3F (high speed)							

Figure 14b demonstrates the scheme used to keep the tracking object in the center of the FOV. Our FOV was divided into 9 regions corresponding to the directions of the pan–tilt. To make the target object size larger or smaller, zoom-out and zoom-in were also used. Every region has a specific direction as shown in Figure 14b. If the target is located on the stop-region, then the camera is set to stop. Meanwhile, the camera speed on other regions is determined by the PID controller. The zoom-in and zoom-out will be activated if the target's size becomes smaller or larger than the user's defined size. The details of the camera control are shown in Figure 15.

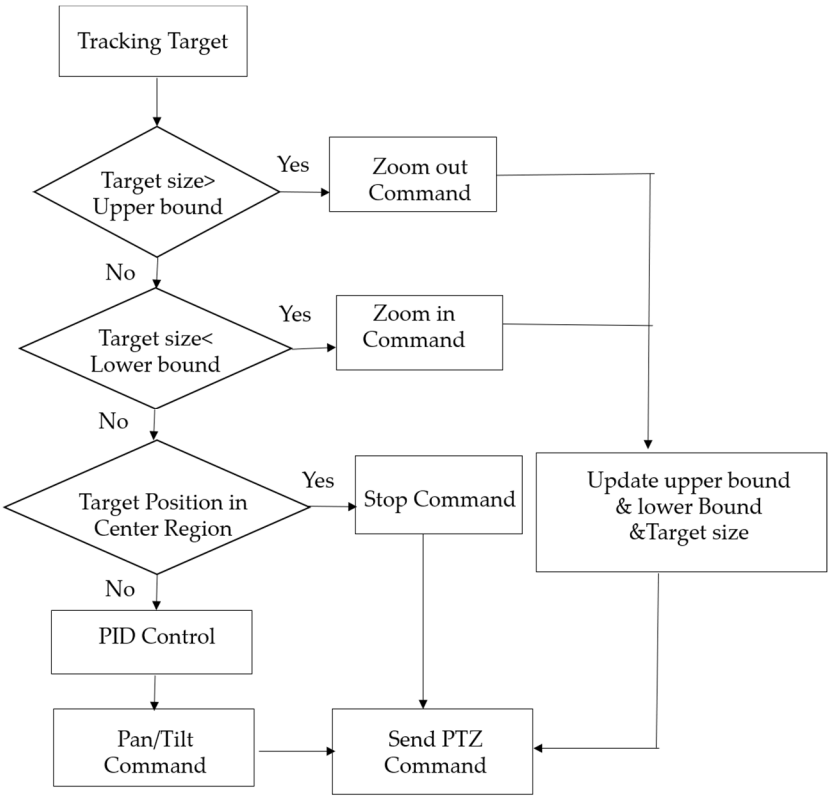


Figure 15. Camera control flow chart.

Our PID controller defines its variables as follows:

- Setting value $u(t)$: Central image position.
- Error signal $e(t)$: Difference between the target position and central position.
- Measured value $y(t)$: Tracking system's estimate of target position.
- Output signal C_{out}: Output value used to control pan/tilt direction and speed.

To control the vertical and horizontal position difference, two independent PID controllers were used. Equations (28) and (29) used C_{out} to estimate the pan/tilt speed, where we defined $offset_{pan}$ and $offset_{tilt}$ values.

$$Speed_{pan} = C_{out} * 0.1 + offset_{pan} \quad (28)$$

$$Speed_{tilt} = C_{out} * 0.1 + offset_{tilt} \tag{29}$$

$$offset_{pan} = \begin{cases} offset_{pan}, & C_{out} \geq 0 \\ -offset_{pan}, & C_{out} < 0 \end{cases} \tag{30}$$

$$offset_{tilt} = \begin{cases} offset_{tilt}, & C_{out} \geq 0 \\ -offset_{tilt}, & C_{out} < 0 \end{cases} \tag{31}$$

The pan and tilt speed of the camera are provided by the manufacturer of the camera (0 to 64). Equations (28) and (29) of PID controller will give the speed in limited range. If the speed is too low, the target object could go out of the frame of the camera by the time the camera moves. On the other hand, if the speed is too high, the camera could lose track of the target object and drive over it.

Depending on the size of the ROI, we decided on whether to zoom in or out. We applied Equations (32) and (33), where we set $rate_{big} = 1.1$ and $rate_{small} = 0.9$, and $w_{initial}$ and $h_{initial}$ were, respectively, the width and height of our human target object.

$$\begin{cases} upper_w = w_{initial} * rate_{big} \\ upper_h = h_{initial} * rate_{big} \end{cases} \tag{32}$$

$$\begin{cases} lower_w = w_{initial} * rate_{small} \\ lower_h = h_{initial} * rate_{small} \end{cases} \tag{33}$$

Upon zoom-in/out, we updated the size of the target model by an aspect of $ratio_{w/h}$, which Equation (34) defines.

$$ratio_{w/h} = \frac{w_{initial}}{h_{initial}} \tag{34}$$

We updated the target model size with Equations (35) and (36) in the case of a zoom-in operation or Equations (37) and (38) in the case of a zoom-out operation. Later, we used these renewed states to update the variables from Equations (32) and (33).

$$\mathrm{w}_{new} = \begin{cases} lower_w * rate_{big} & , if\ \mathrm{w} < lower_w \\ lower_h * rate_{big} * ratio_{w/h} & , if\ \mathrm{h} < lower_h \end{cases} \tag{35}$$

$$\mathrm{h}_{new} = \begin{cases} lower_w * rate_{big} * \frac{1}{ratio_{w/h}} & , if\ \mathrm{w} < lower_w \\ lower_h * rate_{big} & , if\ \mathrm{h} < lower_h \end{cases} \tag{36}$$

$$\mathrm{w}_{new} = \begin{cases} upper_w * rate_{small} & , if\ \mathrm{w} > upper_w \\ upper_h * rate_{small} * ratio_{w/h} & , if\ \mathrm{h} > upper_h \end{cases} \tag{37}$$

$$\mathrm{h}_{new} = \begin{cases} upper_w * rate_{small} * \frac{1}{ratio_{w/h}} & , if\ \mathrm{w} > upper_w \\ upper_h * rate_{small} & , if\ \mathrm{h} > upper_h \end{cases} \tag{38}$$

4. Experimental Results

The proposed method was implemented on a PC platform with Intel® Core™ i5 CPU 650 at 3.20GHz, 4GB RAM, and developed in Borland C++ Builder 6.0 on Windows 7. To verify the performance and stability of the system, it was tested under several environments. We tested both image sequences and video files (AVI uncompressed format) from the active camera, with a resolution of 720 × 480 pixels.

4.1. Results of Tracking on Video File

To verify the tracking algorithm with the proposed particle filter, we used three video files, with parameters as follows:

- Number of bins in histogram $m = 6 * 6 * 6 = 216$
- Number of samples $N = 30$
- State covariance $\left(\sigma_x, \sigma_{v_x}, \sigma_y, \sigma_{v_y}, \sigma_w, \sigma_h\right) = (2,\ 0.5,\ 2,\ 0.5,\ 0.4,\ 0.8)$

Video 1 shows our system's occlusion handler in operation. Figure 16 shows the tracking system without the occlusion handler while Figure 17 shows the same track with our occlusion handler solution. We used the second video to verify the tracking feature. The full occlusion condition happens in frame 3 of Figures 16 and 17. If the particle filter resamples during the full occlusion condition, it may resample on incorrect positions as shown in frame 4, and tracking will be lost, as in frames 5 and 6. Meanwhile, when the full occlusion happens in the particle filter with occlusion handle, the resample step will not be done immediately. Thus, the sample set can keep the widespread range to track the target after full occlusion. 2. Video 2 is used to verify the tracking feature. Figure 18 shows a human wearing a black jacket while walking near a black chair, which is used as an object with similar color features as the human. In this case, the target human has a similar color feature with the black chair, but the proposed system can still track the target human. Video 3 is used to verify the tracking performance in a complex situation. Figure 19 shows the target human is partially occluded with a chair. The target human performs sitting-down and standing-up activities, and later, another human object partially occludes our original target, which continues to be the target, hence showing the system not losing track of the target.

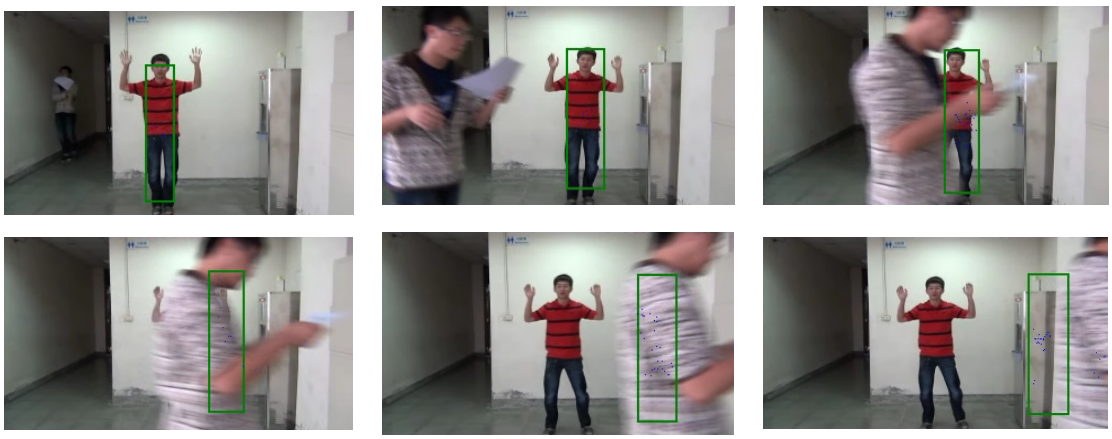


Figure 16. Tracking without occlusion handler.

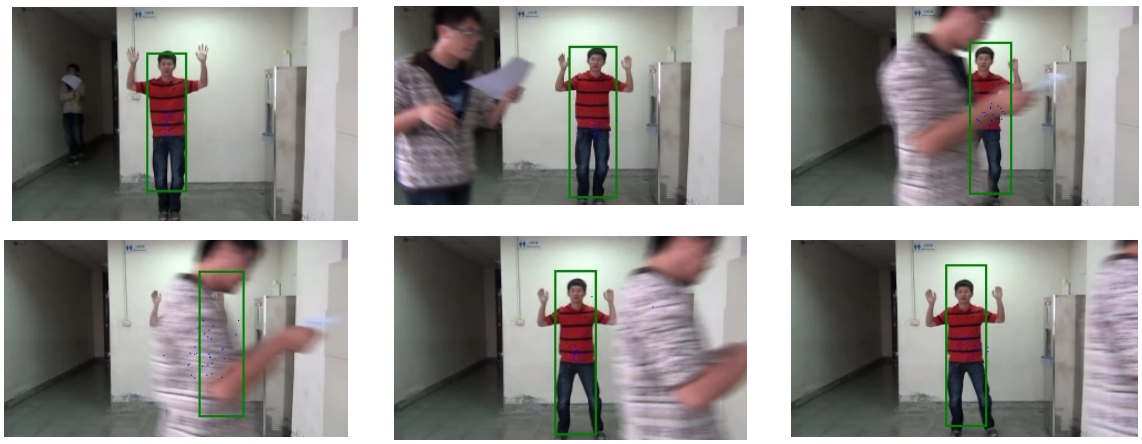


Figure 17. Tracking with occlusion handler.

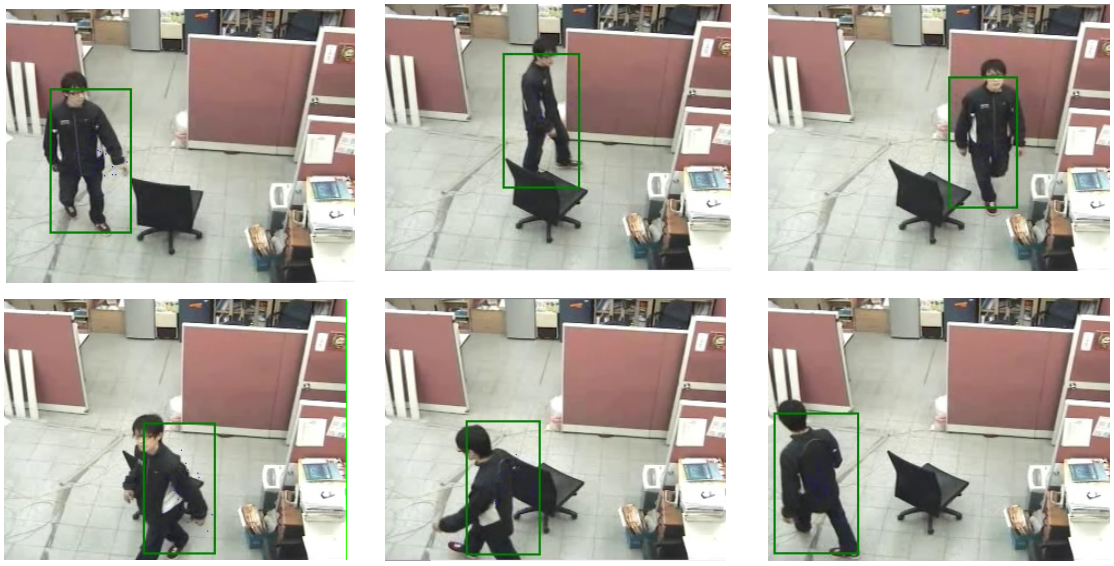


Figure 18. Object has similar color features as the target.

Figure 19. Tracking system performing in a complex situation.

4.2. Results of Tracking on Active Camera Output

We used an active camera set up in our lab, with an environment complex enough to verify the system operation. We set the particle filter and PTZ parameters as follows:

- Number of bins in histogram $m = 6 \times 6 \times 6 = 216$
- Number of samples $N = 30$
- State covariance $\left(\sigma_x, \sigma_{v_x}, \sigma_y, \sigma_{v_y}, \sigma_w, \sigma_h\right) = (10, 1, 10, 1, 1, 2)$
- $\text{Offset}_{\text{pan}} = 12$
- $\text{Offset}_{\text{tilt}} = 6$
- Proportional constant $K_p = 0.9$

- Integral constant K_t =0.1
- Derivative constant K_D = 0.15
- rate_{big} = 1.1
- $\text{rate}_{\text{small}}$ = 0.9

Figure 20 shows the tracking system controlling the pan/tilt of the camera. The targeted human was mostly located in the camera's FOV. Figure 21 shows the results of the zoom in/out while tracking. Figure 22 shows the tracking system controlling the pan/tilt/zoom of the camera, with the targeted human freely walking in the environment. Figures 23 and 24 show our system tracking a target human with more than one person walking in the same environment. While in the test from Figure 23 we see the target only walking around, in the test from Figure 24 we see the human target also performing some more actions, such as crouching and intentionally occluding himself.

Figure 20. Tracking system controlling the pan/tilt of the camera. (**a**) $zoom_{layer}$: 0; (**b**) $zoom_{layer}$: 0; (**c**) $zoom_{layer}$: 1; (**d**) $zoom_{layer}$: 1; (**e**) $zoom_{layer}$: 1; (**f**) $zoom_{layer}$: 0; (**g**) $zoom_{layer}$: 0; (**h**) $zoom_{layer}$: 1; (**i**) $zoom_{layer}$: 2; (**j**) $zoom_{layer}$: 1; (**k**) $zoom_{layer}$: 0; (**l**) $zoom_{layer}$: 0.

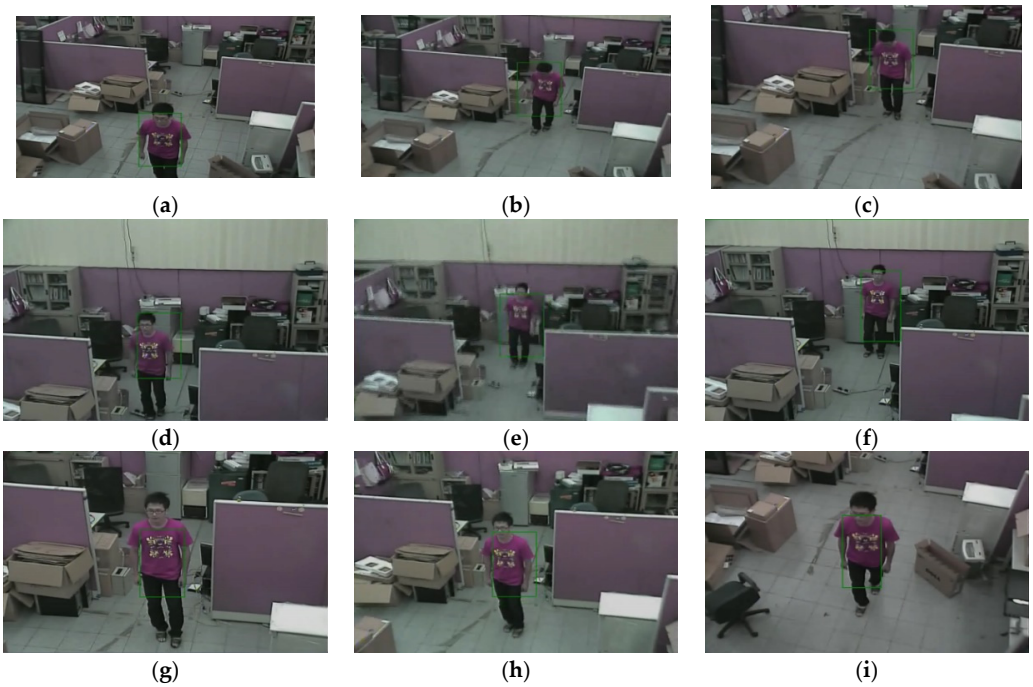


Figure 21. Tracking system performing zoom in/out. (**a**) $zoom_{layer}$: 0; (**b**) $zoom_{layer}$: 0; (**c**) $zoom_{layer}$: 1; (**d**) $zoom_{layer}$: 2; (**e**) $zoom_{layer}$: 1; (**f**) $zoom_{layer}$: 2; (**g**) $zoom_{layer}$: 3; (**h**) $zoom_{layer}$: 2; (**i**) $zoom_{layer}$: 1.

Figure 22. Tracking system performing PTZ.

Figure 23. Human tracking with multiple objects.

Figure 24. Human tracking with multiple objects.

Figure 21a shows the target human has been detected and the $Zoom_{layer}$ is initialized to 0. The targeted human was walking away or approaching the camera. If there is a zoom-in happening, $zoom_{layer}$ is added by 1. On the other hand, $zoom_{layer}$ is subtracted by 1 when zoom-out happens. The details of $zoom_{layer}$ is showed in Tables 2 and 3 for Figures 20a–l and 21a–i respectively.

Table 2. Zoom layer varies in Figure 20.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
$Zoom_{layer}$	0	0	1	1	1	0	0	1	2	1	0	0

Table 3. Zoom layer varies in Figure 21.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
$Zoom_{layer}$	0	0	1	2	1	2	3	2	1

The experiment results show that the proposed system can track a moving human target by particle filter algorithm on an active camera. In addition, the tracking system is able to track the target human when more than one person is walking in the same environment. Moreover, the zoom-in/out adjusts the resolution image while tracking the human. There are several contributions in this research:

1. Our system can accurately distinguish human and nonhuman.
2. The weighted resampling can help the particle filter to preserve the samples with high weights.
3. The occlusion handler can solve the temporal full occlusion condition.
4. It can track the human target smoothly by using the PID controller to determine the motion of camera.

5. Conclusions

In this paper, we proposed a new system that smoothly tracks the human target by camera motion by means of PID controller. The experimental results demonstrated that the proposed system was capable of tracking a moving human target using a particle filter on an active camera. It was also able to precisely differentiate nonhuman and human. In the case when multiple people are walking in the same environment, the tracking system accurately tracked the human targeted. The resolution image of the tracked human can be adjusted using zoom in/out. The weighted resampling used in this paper helps the particle filter to preserve high weight samples. In addition, the temporal full occlusion condition was solved using occlusion handler.

Author Contributions: Conceptualization, L.C.C., D.L.L. and M.P.; methodology, L.C.C., S.P. and M.P.; software, D.J. and M.S.M.; validation, M.S.M., A.S., D.J. and C.T.L.; formal analysis, L.C.C., S.P., D.L.L., A.S. and M.P.; investigation, M.S.M., A.S., D.L.L. and C.T.L.; resources, L.C.C., S.P., M.S.M., D.J. and M.P.; data curation, L.C.C., D.J. and A.S.; writing—original draft preparation, L.C.C. and S.P.; writing—review and editing, L.C.C., S.P., M.S.M., D.J., D.L.L., A.S., M.P., and C.T.L.; visualization, S.P., M.S.M., D.J. and A.S.; supervision, D.L.L. and C.T.L.; project administration, D.L.L. and M.P.; funding acquisition, D.L.L., M.P. and C.T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Australian Research Council (ARC) under Grant DP180100670 and Grant DP180100656, in part by the U.S. Army Research Laboratory under Agreement W911NF-10-2-0022, and in part by the Taiwan Ministry of Science and Technology under Grant MOST 106-2218-E-009-027-MY3 and MOST 108-2221-E-009-120-MY2.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gillner, W.J. Motion based vehicle detection on motorways. In Proceedings of the Intelligent Vehicles '95 Symposium (IEEE), Detroit, MI, USA, 25–26 September 1995.
2. Batavia, P.H.; Pomerleau, D.A.; Thorpe, C.E. Overtaking vehicle detection using implicit optical flow. *Comput. Stand. Interfaces* **1999**, *20*, 466. [CrossRef]
3. Zhao, L.; Thorpe, C.E. Stereo-and neural network-based pedestrian detection. *IEEE Trans. Intell. Transp. Syst.* **2000**, *1*, 148–154. [CrossRef]
4. Viola, P.; Jones, M.; Snow, D. Detecting Pedestrians Using Patterns of Motion and Appearance. *Int. J. Comput. Vis.* **2005**, *63*, 153–161. [CrossRef]
5. Dimitrijevic, M.; Lepetit, V.; Fua, P. Human body pose detection using Bayesian spatio-temporal templates. *Comput. Vis. Image Underst.* **2006**, *104*, 127–139. [CrossRef]
6. Montabone, S.; Soto, Á. Human detection using a mobile platform and novel features derived from a visual saliency mechanism. *Image Vis. Comput.* **2010**, *28*, 391–402. [CrossRef]
7. Pang, Y.; Yuan, Y.; Li, X.; Pan, J. Efficient HOG human detection. *Signal Process.* **2011**, *91*, 773–781. [CrossRef]
8. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*; Addison-Wesley: New York, NY, USA, 1992.
9. Marr, D.; Hildreth, E. Theory of edge detection. *Proc. R. Soc. London. Ser. B Boil. Sci.* **1980**, *207*, 187–217. [CrossRef]
10. Li, R.; Tian, T.-P.; Sclaroff, S.; Yang, M.-H. 3D Human Motion Tracking with a Coordinated Mixture of Factor Analyzers. *Int. J. Comput. Vis.* **2009**, *87*, 170–190. [CrossRef]
11. Lopes, N.V.; Couto, P.A.; Jurio, A.; Melo-Pinto, P. Hierarchical fuzzy logic based approach for object tracking. *Knowl. -Based Syst.* **2013**, *54*, 255–268. [CrossRef]
12. Williams, O.; Blake, A.; Cipolla, R. Sparse Bayesian learning for efficient visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1292–1304. [CrossRef]
13. Collins, R.T.; Liu, Y.; Leordeanu, M. Online selection of discriminative tracking features. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1631–1643. [CrossRef] [PubMed]
14. Zhang, W.; Shang, L.; Chan, A.B. A Robust Likelihood Function for 3D Human Pose Tracking. *IEEE Trans. Image Process.* **2014**, *23*, 5374–5389. [CrossRef] [PubMed]
15. Zhao, H.; Xiang, K.; Cao, S.; Wang, X.-Y. Random walks colour histogram modification for human tracking. *IET Comput. Vis.* **2016**, *10*, 842–851. [CrossRef]
16. Fukunaga, K.; Hostetler, L. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Inf. Theory* **1975**, *21*, 32–40. [CrossRef]
17. Comaniciu, D.; Ramesh, V.; Meer, P. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 564–577. [CrossRef]
18. Wang, L.; Yan, H.; Wu, H.-Y.; Pan, C. Forward–Backward Mean-Shift for Visual Tracking With Local-Background-Weighted Histogram. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 1480–1489. [CrossRef]
19. Vojir, T.; Noskova, J.; Matas, J. Robust scale-adaptive mean-shift for tracking. *Pattern Recognit. Lett.* **2014**, *49*, 250–258. [CrossRef]
20. Vella, F.; Infantino, I.; Scardino, G. Person identification through entropy oriented mean shift clustering of human gaze patterns. *Multimedia Tools Appl.* **2016**, *76*, 2289–2313. [CrossRef]
21. Liu, J.; Zhong, X. An object tracking method based on Mean Shift algorithm with HSV color space and texture features. *Clust. Comput.* **2018**, *22*, 6079–6090. [CrossRef]
22. Ali, A.; Jalil, A.; Ahmed, J.; Iftikhar, M.A.; Hussain, M. Correlation, Kalman filter and adaptive fast mean shift based heuristic approach for robust visual tracking. *Signal Image Video Process.* **2014**, *9*, 1567–1585. [CrossRef]
23. Jeong, J.; Yoon, T.S.; Park, J.B. Mean shift tracker combined with online learning-based detector and Kalman filtering for real-time tracking. *Expert Syst. Appl.* **2017**, *79*, 194–206. [CrossRef]
24. Bhat, P.G.; Subudhi, B.N.; Veerakumar, T.; Laxmi, V.; Gaur, M.S. Multi-Feature Fusion in Particle Filter Framework for Visual Tracking. *IEEE Sens. J.* **2020**, *20*, 2405–2415. [CrossRef]
25. Nummiaro, K.; Koller-Meier, E.; Van Gool, L. An adaptive color-based particle filter. *Image Vis. Comput.* **2003**, *21*, 99–110. [CrossRef]
26. Alcantarilla, P.F.; Bartoli, A.; Davison, A.J. KAZE features. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 214–227.

27. Murray, D.; Basu, A. Motion tracking with an active camera. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 449–459. [CrossRef]
28. Karamiani, A.; Farajzadeh, N. Optimal feature points for tracking multiple moving objects in active camera model. *Multimedia Tools Appl.* **2015**, *75*, 10999–11017. [CrossRef]
29. Lisanti, G.; Masi, I.; Pernici, F.; Del Bimbo, A. Continuous localization and mapping of a pan–tilt–zoom camera for wide area tracking. *Mach. Vis. Appl.* **2016**, *27*, 1071–1085. [CrossRef]
30. Mathivanan, A.; Palaniswamy, S. Efficient fuzzy feature matching and optimal feature points for multiple objects tracking in fixed and active camera models. *Multimed. Tools Appl.* **2019**, *78*, 27245–27270. [CrossRef]
31. Fan, J.; Xu, W.; Wu, Y.; Gong, Y. Human tracking using convolutional neural networks. *IEEE Trans. Neural Netw.* **2010**, *21*, 1610–1623. [PubMed]
32. Tyan, V.; Kim, D. Convolutional Neural Network with Particle Filter Approach for Visual Tracking. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 693–709. [CrossRef]
33. Ren, Z.; Yang, S.; Zou, F.; Yang, F.; Luan, C.; Li, K. A face tracking framework based on convolutional neural networks and Kalman filter. In Proceedings of the 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017.
34. Angelico, J.; Wardani, K.R.R. Convolutional neural network using kalman filter for human detection and tracking on RGB-D video CommIT. *Commun. Inform. Technol. J.* **2018**, *12*, 105–110.
35. Luo, Y.; Yin, D.; Wang, A.; Wu, W. Pedestrian tracking in surveillance video based on modified CNN. *Multimed. Tools Appl.* **2018**, *77*, 24041–24058. [CrossRef]
36. Xia, Y.; Qu, S.; Goudos, S.; Bai, Y.; Wan, S. Multi-object tracking by mutual supervision of CNN and particle filter. *Pers. Ubiquitous Comput.* **2019**, 1–10. [CrossRef]
37. Choe, G.; Choe, C.; Wang, T.; So, H.; Nam, C.; Yuan, C. Deep learning with particle filter for person re-identification. *Multimed. Tools Appl.* **2018**, *78*, 6607–6636. [CrossRef]
38. Aslan, M.F.; Durdu, A.; Sabanci, K.; Mutluer, M.A. CNN and HOG based comparison study for complete occlusion handling in human tracking. *Measurment* **2020**, *158*, 107704. [CrossRef]
39. Stauffer, C.; Grimson, W. Adaptive background mixture models for real-time tracking. In Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, USA, 23–25 June 1999.
40. Salazar, A.; Igual, J.; Safont, G.; Vergara, L.; Vidal, A. Image Applications of Agglomerative Clustering Using Mixtures of Non-Gaussian Distributions. In Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence (CSCI), Washington, DC, USA, 7–9 December 2015.