The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Real Time Tracking using an Active Pan-Tilt-Zoom Network Camera

Thang Dinh, Qian Yu, Gerard Medioni

Abstract— **We present here a real time active vision system on a PTZ network camera to track an object of interest. We address two critical issues in this paper. One is the control of the camera through network communication to follow a selected object. The other is to track an arbitrary type of object in real time under conditions of pose, viewpoint and illumination changes. We analyze the difficulties in the control through the network and propose a practical solution for tracking using a PTZ network camera. Moreover, we propose a robust real time tracking approach, which enhances the effectiveness by using complementary features under a two-stage particle filtering framework and a multi-scale mechanism. To improve time performance, the tracking algorithm is implemented as a multi-threaded process in OpenMP. Comparative experiments with state-of-the-art methods demonstrate the efficiency and robustness of our system in various applications such as pedestrian tracking, face tracking, and vehicle tracking.**

I. Introduction

Based on open IP standards, a network camera can connect to any kind of IP network, including the Internet, and enables remote viewing and recording from anywhere in the world. This provides a promising solution for video surveillance in terms of easy deployment and flexibility in building a multi-camera system. A large number of fixed CCD cameras may be needed to cover an area. By contrast, just a few active PTZ cameras can cover the same area. In order to release human operators from monitoring and controlling these surveillance cameras, real time autonomous control of a PTZ network camera is required. Automatic visual tracking on a PTZ network camera is challenging. Firstly, tracking itself is difficult when there exists apparent pose, illumination and viewpoint changes. Secondly, control over network needs to take care of many practical issues, including communication delay, packet loss and response time. Considering tracking and control together, the requirement of real time performance of tracking is strict. Recently, some methods have been proposed in different applications using PTZ camera such as face tracking, humans and vehicle tracking, driving assistance by tracking obstacles (*i.e* rear view of target vehicles) with a mounted PTZ camera on car. The face tracking system [8] uses template matching, which is poor in dealing with cluttered background or abrupt changes in object appearance. In the humans and vehicles tracking system [9], an integral histogram method is adopted to compute color signature of object, which makes it impossible to overcome the problem of abrupt motion, viewpoint and illumination changes. The framework in driving assistance [10] needs a learning stage (which takes about 7 seconds) and does not update the model during tracking, which causes the inability to adapt to appearance changes.

All of the authors are with Institute for Robotics and Intelligent Systems, University of Southern California, USA {thangdin,qianyu,medioni}@usc.edu

We propose a novel real time tracker for any previously unknown object using a PTZ Network Camera as an input with multi-threading mechanism in receiving, processing, controlling, displaying and saving results. The tracker is robust against abrupt motion and changes in viewpoint and illumination. Our first contribution is a novel framework to overcome the issues in controlling an off-the-shelf PTZ Network Camera while maintaining the quality of service in various real applications. Our second contribution is a robust tracker: a two-stage particle filter and multi-scale mechanism deals with large motion caused by concurrent movements of object and camera, and the delay in communication over the network. It also uses complementary features to enhance its power.

Figure 1 illustrates the overview of our approach, in which the initialization is provided either of two ways: manual or automatic (using detectors when the type of object is previously known). The control part handles all of the issues from control of the camera over the network to maintainance of the quality of service. Finally, the core tracker is a robust real time one with the ability in tracking an arbitrary object. The rest of this paper is organized as follows. The control part of our approach is presented in section II. Details of the tracker are discussed in section III. The experiments and applications are described in section IV, followed by conclusions, future work, and acknowledgements.

II. Control of a PTZ network camera

There are many good approaches to follow objects with an active camera optimally and state-of-the-art theories on adaptive control of time delay systems have been proposed [11]–[14] . However, such approaches need highly-configured devices which can be accessed through low-level interfaces while our goal is tracking via a typical commercial PTZ Camera with limited support (as described below in the description of the camera). Therefore, we adopt the simple but efficient proportional control with a time-sharing strategy to overcome the time delay of the system.

Communication between the control center and the PTZ camera is via TCP/IP network, *i.e*. all captured images and control commands are sent through the network.

A. Description of the camera

We use the off-the-self Sony PTZ Network Camera SNC-RZ30N. This camera offers a large focal length range and

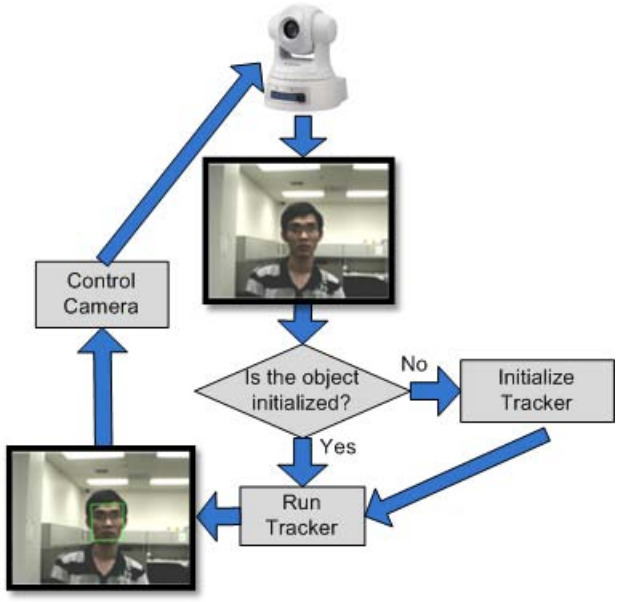
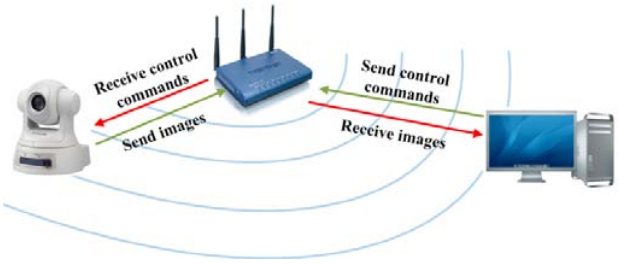


Fig. 1. Our proposed active system



(a) Experimental setup

Zoom Ratio	25x optical zoom, 300x with digital zoom
Horizontal viewing angle	2.0 degrees to 45 degrees
Focal length	f = 2.4 mm to 60 mm
F-number	F1.6 (wide), F2.7 (tele)
Minimum object distance	Tele: 800 mm Wide: 30 mm

(b) Lens specifications

Pan angle	-170 to +170 degrees
Pan speed	2 sec./340 degrees
Tilt angle	-90 to +25 degrees
Tilt speed	1.5 sec./115 degrees

(c) Pan/Tilt specifications

Fig. 2. Experimental setup and specifications from manual

wide field-of-view. Another advantage of the camera is the simplicity of installation. The camera protocol is popular and compatible with many other types of SONY Network cameras. This makes the system easy to extend to multiple cameras. The camera specifications and the configuration of our experiment setup are shown in Figure 2. The wireless connection is established through a Cisco Aironet 350 series adapter.

Although the network camera (especially through wireless network) is easy to deploy and communicate, proper control is very difficult. Due to the nature of network communication, there is a delay in receiving acquired frames and sending control commands. Also, the overhead occurred in getting the camera status, *e.g.* pan-tilt pose and focal length, is expensive in terms of time constraints in real-time systems. In order to handle this problem, a time-sharing strategy is introduced to balance the load of sending control commands and inquiring the camera status. Another issue in the control part is that the camera provides a discrete speed control due to the use of stepping motors, *i.e.* 14 levels of speed control for tilt and 18 levels for pan. Therefore, we first calibrate the pan-tilt speed at each level and store this information in a lookup-table.

B. Formulation

In the PTZ camera model, let O, C represent the optical center and the principal point respectively. f is the focal length of the camera. The center of the target is projected at P in the image plane. The size of the projection of the target is represented as S. Intuitively, the goal of the tracking is to

1) keep the target in the center of the image.
2) keep the projection of the target at a proper size.

Pan and tilt are defined in camera coordinates along the X, Y axes respectively. The first goal can be achieved by controlling pan and tilt, while the second one can be obtained by changing the focal length f. Suppose θ is the angle between the ray passing through the target center and the optical axis and S_0 is the desired size of the projection of the target in the image plane, the goal of control is

$$\begin{cases} S \to S_0 \\ \theta \to 0 \end{cases} \tag{1}$$

θ is determined by two factors: $\theta = \arctan \frac{||P-C||}{f}$. The numerator is the distance between the projection of the target center P and the principal point C (note that C is the image center which is used as an approximation). The goal is to minimize the distance between P and C with respect to the current focal length. For instance, when f is large (zoom-in on the target) a small pan-tilt motion drastically changes the offset in the image plane.

C. Control

Since a request over the network has large communication delay, we cannot acquire accurate camera pose information at every frame. The only way to control the network camera is by sending the command to change the motion direction and speed of the camera. In other words, the camera status can be changed by altering the derivatives of the state $S(t)$, *i.e.*

$$S(t+1) = S(t) + \dot{S}(t) \tag{2}$$

where $S(t)$ represents the control variable, including pan, tilt and zoom $S(t) \triangleq [pan(t), tilt(t), zoom(t)]$. The target

of interest may undergo continuous generic 3D motion and the tracking result may contain errors. Considering these two practical conditions in tracking, we adopt the P control, where the correcting term is proportional to the current error value. Let $u(t)$ represents the input to the PTZ control process,

$$u(t) = K_p e(t) \tag{3}$$

where K_p is the proportional gain and $e(t)$ is the current error term. We adopt the following proportional term,

$$\dot{S}(t) = \left[K_1 \frac{||P-C||_X}{f}, K_2 \frac{||P-C||_Y}{f}, K_3 \left(\frac{S}{S_0} - 1 \right) \right] \tag{4}$$

where $||P-C||_X$ and $||P-C||_Y$ represent the offset between P and C along X and Y axes in the camera coordinates. Since there is no integral term in Eq.4, this may cause system oscillation. In practice, the integral term may make a slow response to the whole system due to the target undergoing continuous motion. Therefore, a truncated negative feedback is used to avoid oscillation and achieve a fast response, namely

$$\begin{aligned} \dot{S}(t) = \Big[& K_1 \delta \left(\tfrac{||P-C||_X}{f}, \lambda_1 \right), K_2 \delta \left(\tfrac{||P-C||_Y}{f}, \lambda_2 \right), \\ & K_3 \delta \left(\tfrac{S}{S_0} - 1, \lambda_3 \right) \Big] \end{aligned} \tag{5}$$

where the function $\delta(\cdot, \lambda)$ is a truncation function,

$$\delta(x, \lambda) = \begin{cases} x, x \geq \lambda \\ 0, x < \lambda \end{cases} \tag{6}$$

In Eq.4, we need to know the current focal length to determine proper pan and tilt speed. However, frequent inquiry commands increase communication overhead and cannot be achieved on the current network camera. Fortunately, the focal length of the camera does not change drastically compared to the pan and tilt speed. Thus, we do not request the focal length at every frame.

There are three types of control commands defined in this system. One type is to change the pan and tilt speed, which is rated as high priority to make sure the camera can follow the target.The second type is to change the current focal length and the third type is to inquire the current focal length, which is given low priority since focal length should not change often. The PTZ camera has individual modules to respond to each type of commands. None of these commands can be sent too frequently, otherwise the command queue fills up and later commands are ignored. In order to manage to communicate with the camera by using these three types of commands, a time-sharing strategy is applied. A number of Pan-Tilt commands and Zoom commands form a sending group. Several sending groups with one inquiry command then form one control loop. The frequency of each type of command is determined by its weight. In our experiments, we adopt one Pan-Tilt command and one Zoom command as one sending group, and two sending groups with one inquiry command form a control loop. Note that we cannot acquire the absolute pan, tilt or zoom information by accumulating the increments with initial status. The command sent to the camera is achieved by mechanical processes, which may contain some errors. These errors are difficult to estimate or to predict. Accumulating the increments produces a large accumulated error. The only way to acquire the camera status is by request.

D. Validation

In the API package of the SONY camera, there are two functions which help to pan/tilt and zoom to a specified position and area, respectively: directPT(x,y) and areaZoom(x,y,w,h). We adopt a simple and fast tracker to follow a face and keep it always in a proper size. The camshift tracker with skin color model is applied because of its extremely fast running speed. However, these functions are only applicable when dealing with static objects. By mentioning static objects we want to emphasize that as long as the object is moving, those two functions cannot track the current position/size of the object. By the time they try to pan/tilt or zoom to the preferred region, the target has moved far away. Therefore, instead of using these high-level supporting functions, we use the primitive APIs to follow the object. In section IV, several challenging situations are shown to demonstrate the robustness of our control and tracker in different settings.

III. Tracking

Object visual tracking is a very broad research area. There is a strong relationship between object representations and tracking algorithms. Also, selecting informative features that describe the attributes of objects is an important issue. Different features represent different complexities and tracking outcomes. An objects can be represented as a set of feature points with a specific structure [5] which is an efficient representation but it needs high resolution input and lacks of other important information such as color and pattern. A contour can also be used to represent an object when its boundary is more stable such as in [6]. An object can also be represented as a segmentation, which provides the most accurate, pixel-level representation of an object in the 2D image. Given such a representation, one can always switch to any other simpler representation, like point, shape and contour. Moreover, it can be used to represent a large variety of objects: rigid or non-rigid objects, simple shape or complex shape objects. Several tracking methods based on pixel-level segmentation have been proposed such as [15]. However, both contour and segmentation based representation need a good initialization for tracking an arbitrary object, which is not applicable in our system. The commonly used features for visual tracking include color, local features (point, curve, patch), and global pattern. Color is invariant to camera motion and scale changes. Successful color trackers include [16], [17]. However, color is not always available, e.g. in IR videos and may be distracted by background. Local features such as [1], [18] can provide an efficient representation and accurate localization; however, it needs enough resolution of input sequence for robustness. Pattern encodes the global appearance of objects like in

(a) Abrupt motion in two consecutive frames (b) Motion blur

Fig. 3. Tracking issues with a PTZ Camera (cropped from original images)

[19], [20], but not invariant to camera motion. Thus, we adopt a combination of these three types of features in our system. Due to the active steerable network camera where the concurrent movement of object and camera, and the network latency exist, we often encounter abrupt motion and motion blur (see Figure 3), which is hard to cope with. To address this issue, the idea of using a cascade of particle filters which has been successfully applied in low frame rate video [21] is adopted to propose a two-stage particle filter. In the first stage, a discriminative pre-filter automatically chooses the feature which best separates object and background from feature pool to build a confidence map which helps to remove all bad samples, *i.e.* ones have low confidence. In the second stage, a rough model of object appearance is built at the beginning and continuously updated according to all changes in pose, viewpoint, illumination by encoding them in a low dimension subspace. Moreover, a multi-scale selection is used to improve the coverage of sampling without the need of increasing number of samples. Also, the KLT features [1] are tracked simultaneously to guarantee the consistency of a tracked region. The advantages of this approach are from real time performance with efficient implementation and utilizing the power of complementary features: pixel, local patch, and pattern. Figure 4 shows the overview of our tracker.

A. Probabilistic framework

The tracking problem is formulated as a temporal filter which estimates:

$$P(s_t|F_t) \tag{7}$$

where *t=1, 2, 3...*, $F_t = (f_1, f_2, ..., f_t)$ are the image frames and s_t is the state of the object at time t. We use $s_t = [x, y, \rho_x, \rho_y, \theta, \omega]$ where (x, y) is the center of the tracking box (ρ_x, ρ_y) is the scale regards to the predefined size of the object and θ, ω is the rotation and skew angles, respectively. To avoid drifting, the tracker needs to find the the object with an accurate center position at the right scale, rotation and skew. At frame I_t, the result given by the tracker is a cropped image determined by the state of the tracked object. Assuming a Markovian state transition, we formulate the posterior as a recursive equation:

$$p(s_t|F_t) \propto p(f_t|s_t) \int p(s_t|s_{t-1}) p(s_{t-1}|F_{t-1}) ds_{t-1} \tag{8}$$

where $p(s_{t-1}|F_{t-1})$ is the posterior distribution from all the previous observations while $p(f_t|s_t)$ and $p(s_t|s_{t-1})$ are the observation and transition model respectively. Typically, we

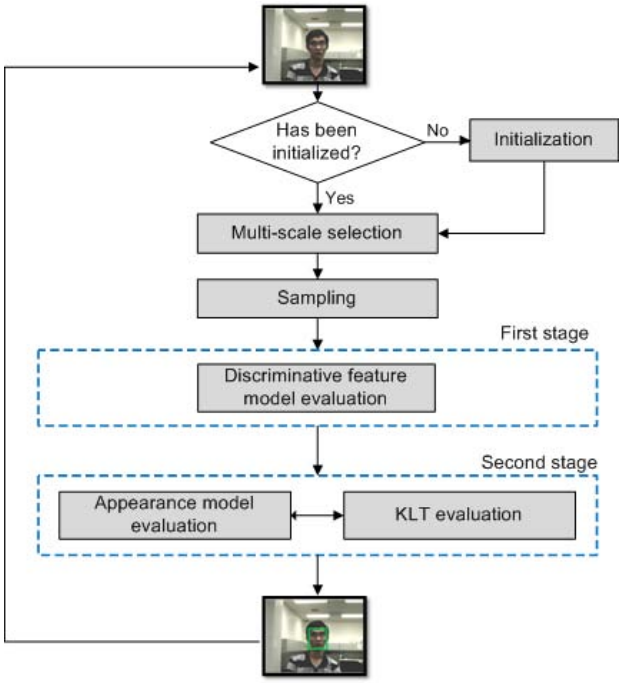


Fig. 4. Overview of the tracker

can assume the smoothness in changing state s_t of the object over time. We assume the transition is Gaussian

$$p(s_t|s_{t-1}) = \mathcal{N}(s_t; s_{t-1}, \Psi_t) \tag{9}$$

where Ψ_t is the time variant diagonal covariance matrix. The recursive inference in Eq.9 is implemented with sampling-based particle filtering [7], in which the conditional density is approximated by maintaining a set of weighted particles.

The critical issue is the estimation of the likelihood of the new observation given the posterior distribution. In our approach, for real time performance, a new observation is evaluated in two stages. In the first stage, the likelihood is computed by calculating the total density from the confidence map of each sample using the discriminative selection feature model; while in the second stage, the confidence is the joint likelihood computed from the object appearance model built by the online learning subspace and the consistency of KLT features maintained in each sample. The tracking result (in the new frame) is the particle with the highest likelihood in the second stage. However, the confidence of samples in the first stage is only used to filter bad samples, while just the values computed from the second one are applied for re-sampling process. Another key part to improve the speed of our system is the parallel evaluation. Because the particles are independent of each other, each one can be evaluated individually. To optimize this process, the evaluation step is implemented in multi-threading manner which can be naturally mapped to a multi-core or distributed system.

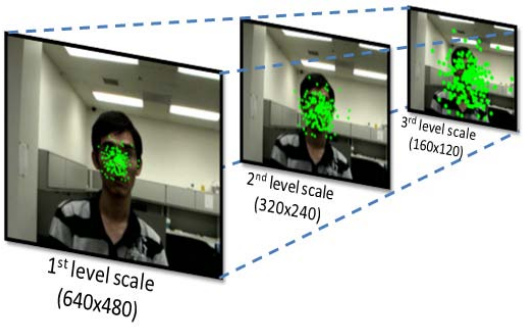


Fig. 5. Multi-scale illustration

B. Multi-scale selection

The risk of missing the target is very high because of abrupt motion, which usually happens when dealing with active cameras. Our case is even more extreme when the network delay issue makes the abrupt motion severe. To address this issue, we propose a multi-scale selection which helps to increase the search range by enlarge the coverage of sampling. The multi-scale approach using belief propagation [4] has been successfully proposed to deal with abrupt motion. However, we only focus on improving the coverage of particle filtering to avoid the need to increase the number of particles in order to maintain the robustness of the tracker. The smaller the scales, the larger the search range can cover. Moreover, because our sample has a relative small pre-defined size (20x20), it is not necessary to keep the input object region large. In our framework, three scale levels: 640x480, 320x240, and 160x120 (see Figure 5) are chosen with respect to the size of the current tracked object.

C. Discriminative feature selection model

In the first stage of our framework, we apply a simple classifier [17] in order to fast discard bad samples before transferring the rest to the second stage. The key idea of this method is to find the feature that best distinguishes between object and background. The feature pool is a set of features composed of linear combinations of R,G,B color with relative weights

$$F \equiv \{a_1 R + a_2 G + a_3 B | a_* \in [-2, -1, 0, 1, 2]\} \quad (10)$$

Given a feature $f_k = \{a_k, b_k, c_k\}$, the histograms $H_{fg}(i)$ and $H_{bg}(i)$ for the pixels on foreground (object) and background are computed, where i denotes the i^{th} bin in the histograms. The log likelihood at the i^{th} bin is given by:

$$L(i) = log \frac{max\{p_{fg}(i), \delta\}}{max\{p_{bg}(i), \delta\}} \quad (11)$$

where δ is a small value (which is set to 0.001) to avoid the division by zero issue while p_{fg} and p_{bg} are the discrete probability densities of the object and background respectively. Then the variance ratio is calculated given the log likelihood function and the probability densities of foreground and background:

$$VR(L; p_{fg}, p_{bg}) \equiv \frac{var(L; (p_{fg} + pbg)/2)}{var(L; p_{fg}) + var(L, p_{bg})} \quad (12)$$

in which the variance of $L(i)$ with respect to p is defined as

$$var(L; p) = \sum_i p(i) L^2(i) - [\sum_i a(i) L(i)]^2 \quad (13)$$

Therefore, intuitively, in Eq.12 and Eq.13, the goal is to maximize the variance between two classes while minimizing the within class ones. The top discriminative feature having highest VR score is chosen to compute the likelihood images (*i.e.* confidence map).

It is important to note that we prefer to apply the feature selection procedure after each 30 frames interval in order to adapt with the appearance change of the object and background.

Approximate Rotated Integral Histogram

In order to utilize the pixel-level information from the discriminative feature selection method within pattern-like framework, in which a rotated window is considered, the accumulated confidence of all pixels in a window is calculated. Moreover, to select the best feature in the discriminative model, a histogram calculation is needed. With respect to real-time performance, to reduce the computational time for evaluating the total confidence of a rotated rectangle, we apply the integral image and approximately decompose that rectangle into union of sub-rectangles with sides parallel to the image coordinates. As shown in Figure 6, after the integral image $\mathcal{I}$ is computed, the total confidence of a region R is estimated as: C(R)=$\Sigma C(r_i)|\Sigma S(r_i) \approx S(R)$, where $S(x)$ is the area of the rectangle x as shown in Figure 6.

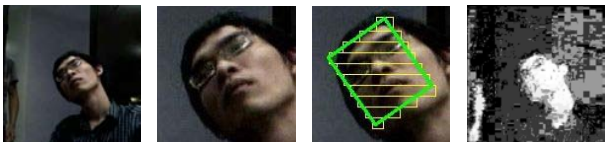
In addition, given the integral image $\mathcal{I}, C(r_i)$ represented as $C(x1, y1, x2, y2)$ is calculated quickly as follows:

$$C(r_i) = \mathcal{I}(x2, y2) - \mathcal{I}(x1, y2) - \mathcal{I}(x2, y1) + \mathcal{I}(x1, y1) \quad (14)$$

The average confidence is then obtained:

$$\bar{C}(R) = \frac{C(R)}{S(R)} \quad (15)$$

In practice, in our system, the object is relatively small compared with image size, we roughly compute the confidence of a rotated area by dividing it into 10 rectangles. This approximation is also applied in estimating the pixels inside any rotated rectangle to compute its histogram.



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

Fig. 6. Rotated histogram rectangle computation and confidence map. From left to right: (a) original image, (b) the cropped face area, (c) the approximated rectangles decomposition, and (d) the final confidence map.

D. Object appearance model

Since the appearance of the object may be different due to the changes of pose, view angle, illumination, *etc*. by time, it is necessary to model all of those variations compactly and precisely. We adopt the power and efficiency of the method proposed in [19], which encodes the appearance of the object into a low-dimension linear subspace and incrementally update it during tracking process.

For initialization, after collecting several samples by simple template matching we train the model of the object from those n training images $\mathcal{I}_{ini} = \{I_1, ..., I_n\}$ by computing the eigenvectors U of the covariance matrix $\frac{1}{n-1}\sum_{i=1}^{n}(I_i - \bar{I})(I_i - \bar{I})^T$, where $\bar{I} = 1\frac{1}{n\sum_{i=1}^{n} I_i}$ is the mean of the training images. And, this can be done by solving the singular value decomposition (SVD) $A = U\Sigma V^T$ of the centered data matrix $[(I_1 - \bar{I})...(I_n - \bar{I})]$

Given new m images $\mathcal{I}_{add} = \{I_{n+1}...I_{n+m}\}$, the subspace needs to be updated by calculating [A B] where B is the new observation matrix according to $\mathcal{I}_{add}$. As the result of derivation in [19], we have

$$[A \quad B] = \left([U \ \tilde{B}]\tilde{U}\right)\tilde{\Sigma}\left(\tilde{V}^T \begin{bmatrix} V^T & 0 \\ 0 & 1 \end{bmatrix}\right) \tag{16}$$

In which $\tilde{B}$ is the component of B orthogonal to U. Finally, we have $U' = [U \ \tilde{B}]\tilde{U}$ and $\Sigma' = \tilde{\Sigma}$ as the updated eigensystem. In our implementation, for efficiency, the top k eigenvectors ($k = 10$) are maintained to represent the model of the learned face.

E. Sample evaluation

1) Appearance model evaluation: Given a subspace Ω with the first k eigenvectors, the projection of x on Ω is y=$(y_1, ..., y_m)^T = U^T(\mathbf{x} - \hat{\mathbf{x}})$. Then the likelihood of x can be expressed as:

$$p(\mathbf{x}|\Omega) = \left[\frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{m}\frac{y_i^2}{\lambda_i}\right)}{(2\pi)^{m/2}\prod_{i=1}^{m}\lambda_i^{1/2}}\right] \cdot \left[\frac{\exp\left(-\frac{\varepsilon^2(\mathbf{x})}{2\rho}\right)}{(2\pi\rho)^{(d-m)/2}}\right] \tag{17}$$

Where λ_i is the eigenvalue with respect to y_i, d is the dimension of the input, $\varepsilon(\mathbf{x}) = |\mathbf{x} - UU^T\mathbf{x}|$ is the projection error. The parameter $\rho = \frac{1}{d-m}\sum_{i=m+1}^{d}\lambda_i$, can be approximated as $\rho = \frac{1}{2}\lambda_{m+1}$.

2) KLT evaluation: KLT features are detected and tracked simultaneously within the object region from the first frame. The confidence of KLT tracker is given by the ratio of features maintained in a tracked region compared to the previous state

$$p_{KLT} = \left(\frac{n_{cur}}{n_{prev}}\right)^2 \tag{18}$$

The final likelihood is the product of the two independent evaluation given in Eq. 17 and Eq. 18. In experiments, to avoid the risk of drift when the object is small and the number of KLT features is limited, the evaluation is only based on appearance model when the number of features is less than a predetermined threshold θ ($\theta = 8$). It is only a rough threshold to keep the system conservative when lacking of local features.

TABLE I

COMPARISON BETWEEN OUR TRACKER AND OTHERS INCLUDING: IVT [19] AND ODT [17] IN DIFFERENT CHALLENGING DATASETS.

	Object type	Frames	IVT	ODT	**Ours**
Seq1	Face (indoor)	427	287	142	**427**
Seq2	Face (indoor)	328	232	328	**328**
Seq3	Face (indoor)	195	80	195	**195**
Seq4	Vehicle(outdoor)	470	470	470	**470**
Seq5	Vehicle(outdoor)	233	233	233	**233**
Seq6	Vehicle (UAV)	235	62	105	**235**
Seq7	Human (outdoor)	868	394	244	**858**

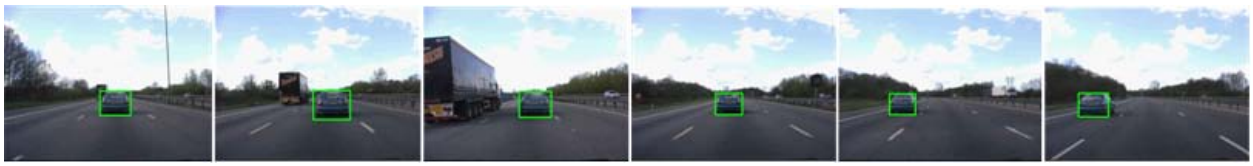
F. Validation

To validate the robustness of our tracker, we compare it with the Incremental Visual Tracker (IVT) [19] [1] and the Online Discriminative Tracker (ODT) [17]. The ODT is implemented in C++ and does not consider large scale change and rotation. It is worth to note that ODT uses kernel-based tracker [3] as a mode-seeking procedure after having the likelihood map in order to find the object in the current image. The parameters are set in both methods and ours are the same. We compare these methods with different challenging situations for different types of object: face (seq1-seq3), vehicle (seq4-seq6), and human (seq7). The data also contain different environments such as indoor (seq1-seq3) and outdoor (seq4-seq7). Seq6 is UAV data of a turning car. The challenging conditions include significant changes in illumination (seq1-seq3, and seq7), abrupt motion (seq1, seq6, and seq7), viewpoint changes and large pose variations (seq1-seq3, seq6, and seq7). Seq1-seq3 are captured in the lab using the PTZ Camera. Seq4-seq5 are from PETS 2001 dataset (Dataset5-Testcamera1, Dataset1-Testcamera1, respectively). Seq6 is drawn from Vivid I dataset, while Seq7 is taken on campus using a handheld camera to follow the object. Except seq4 and seq5, all of other sequences, which are captured from moving cameras, contain large motion blur and dynamic background.

Table 1 shows the comparison result, in terms of the number of frames tracked by each method. The comparison shows that our tracker is more robust than the other methods. Note that the performance of IVT could be improved by increasing the number of particles; however, this makes the method extremely slow.

In the ODT [17], only color information is exploited and no rotation and large scale is considered, which makes the method not robust. For some of the cases (seq2-seq5), although it can follow the object till the end of the sequence, only a part of the object is tracked correctly. During the experiments, we notice that multi-scale helps in increasing the search range of the sampling while the two-layer particle filter framework obtains better coverage by efficiently increasing the number of particles. At the same time, the

[1]The implementation is from http://www.cs.toronto.edu/ dross/ivt/



(a) Rearview vehicle tracking (from PETS 2001, Dataset5)

(b) Sideview vehicle tracking (from PETS 2001, Dataset1)

(c) UAV Data (from VIVID I Dataset)

(d) Pedestrian tracking

Fig. 7. Some results (seq4, seq5, seq6, and seq7) of our tracking approach.

Fig. 8. Indoor face tracking results includes abrupt motion and motion blur while panning/tilting and zooming, with large variations in viewpoint, illumination.

Fig. 9. Outdoor pedestrian tracking in a high-cluttered background includes occlusions, large variations in viewpoint, abrupt motion and motion blur.

KLT plays an important role in getting our tracker on the right track at the beginning while IVT attaches some noise background very soon when the appearance model is still not good enough. Some of our tracking results are shown in Figure 7.

However, the failure cases of our approach happen when there is an abrupt change in object appearance to which has not been observed by the appearance model before.

Our method runs at 35 fps when processing VGA video sequence with a 640x480 resolution; whereas IVT runs at 4.5fps and the ODT runs at 6fps.

IV. Experiments and Applications

A. Experiment details

The experiments are performed using the camera described in Section II and a wireless router and wireless adapters as shown in Figure 2. The approach is implemented in C++ using OpenCV on an Intel Core2duo 3.73GHz with 3GB RAM. The speed is 40 fps (without saving results) when processing VGA video sequences. However, to maintain the quality of service as discussed in the previous section, we set the transfer/display speed to 15 fps. The multi-

threading OpenMP API is used to improve the running speed significantly. In order to reach our real time goal, for sampling, 1500 particles are generated which are then reduced to by 50% after the pre-filtering stage. In this step, the size of the auto-select background region is fixed to three times of the size of the target. In the appearance model, an image vector of 20x20 is applied; 10 eigenvectors are maintained during tracking. After a batch of 5 processed frames, we update the model one time. The number of threads simultaneously processing the sample evaluation step are set to 4, which increases the performance by a factor of 3. In the KLT feature tracker, we use the patch window 10x10 as a candidate feature. In practice, consistency is guaranteed as long as the difference in total energy is less than 40%. The agreement between the evaluation of our appearance model and this KLT tracker is also maintained when over 50% feature points are covered. This is just a soft constraint to avoid some distracters when the model has not been built robustly enough, *i.e.* in the very first frames.

B. Applications

1) Face tracking: We can tag a face or run in automatic mode (using the cascades of boosting classifiers face detector proposed in [2]). All of the appearances of the face are encoded. The results shown in Figure 8 are from a sequence taken in an indoor environment. The difficulties include abrupt motion and motion blur when the control part keeps panning/tilting and zooming to maintain the proper position/size of the object. The variations in viewpoint are also large. This video sequence has 735 frames at a resolution of 640x480.

2) Pedestrian tracking: We tag then follow a walking pedestrian in a large view by pan and tilt. The experiment shown in Figure 9 is taken outdoor. The issues raised here include high-cluttered background with many distracters, abrupt motion, occlusion, and viewpoint changes. The video contains 145 frames with a resolution of 640x480.

For more experimental results, please refer to the supplemental video.

V. Conclusion and future work

We presented a novel tracking approach using a commercial PTZ Network Camera that can be widely applied in many applications because of robustness and real time. The proposed framework also addresses the issues of controlling the camera while maintaining quality of service. The tracking methodology can follow previously unknown objects under variations in viewpoints, illumination and abrupt motion. The approach not only utilizes the power of complementary features such as pixel color, local patch, and pattern but also successfully overcomes the abrupt motion and motion blur under a two-stage particle filters with multi-scale mechanism. The tracking approach fits well with the control part to form a real time system which can track an arbitrary object in multiple environments using the PTZ Network Camera. The tracking approach also produces better results than current state-of-the-art methods. However, the zoom control of PTZ camera is still an open issue which needs to be improved. In the future, we would like to increase the frame rate of the camera by filtering bad images. We are also interested in developing our framework on robotic platforms for more applications.

VI. Acknowledgements

This research was funded, in part, by MURI-ARO W911NF-06-1-0094. The first author was also supported by Vietnam Education Foundation. We thank Eunyoung Kim for helping us to collect data. We also thank Dr. Jongmoo Choi and Himanshu Neema for helpful discussions.

References

[1] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.
[2] P. Viola and M. Jones. Robust real-time object detection. In *IJCV*, volume 57, pages 137–154, 2004.
[3] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. In *PAMI*, volume 25, pages 564–577, 2003.
[4] G. Hua and Y.Wu. Multi-scale visual tracking by sequential belief propagation. In *CVPR*, pages 826–833, 2004.
[5] N. Dowson and R. Bowden. Simultaneous modeling and tracking (smat) of feature sets. In *CVPR*, pages 99–105, 2005.
[6] T. Schoenemann and D. Cremers. Globally optimal shape-based tracking in real-time. In *CVPR*, pages 1–6, 2008.
[7] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. In *IJCV*, volume 29, pages 5–28, 1998.
[8] T. Funahasahi, M. Tominaga, T. Fujiwara, and H. Koshimizu. Hierarchical face tracking by using ptz camera. In *FGR*, pages 427–432, 2004.
[9] M. Lalonde, S. Foucher, L. Gagnon, E. Pronovost, M. Derenne, and A. Janelle. A system to automatically track humans and vehicles with a ptz camera. In *Visual Information Processing XVI*, 2007.
[10] X. Clady, F. Collange, F. Jurie, and P. Martinet. Object tracking with a pan-tilt-zoom camera: application to car driving assistance. In *ICRA*, pages 1653–1658, 2001.
[11] A. P. Detection and T. using Fuzzy Controlled Active Cameras. K. bernadin, f. van de camp, r. stiefelhagen. In *CVPR*, pages 1–8, 2007.
[12] V. K. Singh, P. K. Atrey, and M. S. Kankanhalli. *Coopetitive multi-camera surveillance using model predictive control*, 2008.
[13] M. R. Matausek and A. D. Micic. On the modified smith predictor for controlling a process with an integrator and long dead-time. In *IEEE Transactions on Automatic Control*, volume 41, pages 1199–1203. 1996.
[14] F. C. Teng, G. F. Ledwich, and A.C.Tsoi. Extension of the dahlin-higham controller to multivariable systems with time-delays. In *Int. J. Control*, volume 25, pages 337–350, 1994.
[15] I. Leichter, M. Lindenbaum, and E. Rivlin. Bittracker-a bitmap tracker for visual tracking under very general conditions. In *PAMI*, volume 30, pages 1572–1588, 2008.
[16] K. Nummiaro, E. Koller-Meier, and L. vanGool. Object tracking with an adaptive color-based particle filter. In *DAGM*, pages 353–360, 2002.
[17] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. In *PAMI*, volume 27, pages 1631–1643, 2005.
[18] Z. Kim. Real time object tracking based on dynamic feature grouping with background subtraction. In *CVPR*, pages 1–8, 2008.
[19] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. In *IJCV*, volume 77, pages 125–141, 2008.
[20] X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo. Robust visual tracking based on incremental tensor subspace learning. In *ICCV*, pages 1–8, 2007.
[21] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *CVPR*, pages 1–8, 2007.