# High Resolution Face Sequences from A PTZ Network Camera

Thang Ba Dinh[1], Nam Vo[2] and Gérard Medioni[1]
[1] Institute of Robotics and Intelligent Systems University of Southern California, Los Angeles, CA 90089
{thangdin,medioni}@usc.edu
[2] Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
nam.vongoc@gmail.com

***Abstract*— We propose here to acquire high resolution sequences of a person's face using a pan-tilt-zoom (PTZ) network camera. This capability should prove helpful in forensic analysis of video sequences as frames containing faces are tagged, and within a frame, windows containing faces can be retrieved. The system starts in pedestrian detector mode, where the lens angle is set widest, and detects people using a pedestrian detector module. The camera then changes to the region of interest (ROI) focusing mode where the parameters are automatically tuned to put the upper body of the detected person, where the face should appear, in the field of view (FOV). Then, in the face detection mode, the face is detected using a face detector module, and the system switches to an active tracking mode consisting a control loop to actively follow the detected face with two different modules: a tracker to track the face in the image, and a camera control module to adjust the camera parameters. During this loop, our tracker learns online the face appearance in multiple views under all condition changes. It runs robustly at 15 fps and is able to reacquire the face of interest after total occlusion or leaving FOV. We compare our tracker with various state-of-the-art tracking methods in terms of precision and running time performance. Extensive experiments in challenging indoor and outdoor conditions are also demonstrated to validate the complete system.**

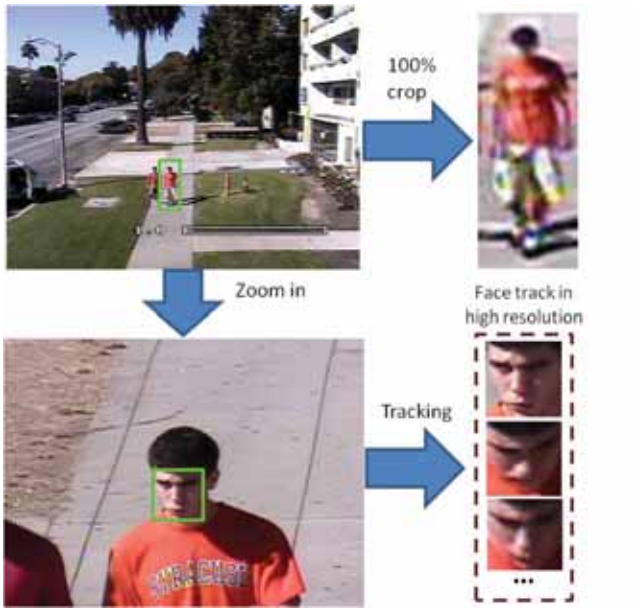


Fig. 1. Scenario

## I. INTRODUCTION

### A. Motivation

Research on human faces is a very important and interesting topic in computer vision with various applications such as biometric identity. Regular CCTV cameras cannot extract human faces with reasonable resolution when they are far away. PTZ cameras can zoom, pan, and tilt to get a close view. Today, this process only occurs under human control, which is impractical with a large number of cameras. Thus, automatically generating high resolution face sequences from PTZ cameras would be very helpful. Many challenges such as network delays, packet loss, and slow response commands still need to be addressed. Given such a scenario, we need not only a robust real-time tracker but also a flexible and smooth control module. The tracker needs to learn the face appearance changes under different conditions. It also has to be robust against cluttered background, abrupt motion, motion blur, and must reacquire the object after total occlusion or leaving FOV. In addition, when a PTZ camera is in wide mode, it performs as a regular CCTV camera, which means when a person is far from the camera, no face detector can detect the face within only few pixels (as shown in Figure 1). A practical system should automatically identify the region of interest containing the face, and zoom to detect, then track this face.

### B. Related work

To address these issues, several methods have been proposed [3], [11], [17], [18]. In [3], Bagdanov *et al*. triggered the system by using OpenCV face detector to find the face of interest and employed KLT [20] features to keep following the object. To locate the object, M-estimator of the centroid of the feature point clouds was used. Even though this method guarantees real-time performance, it is not robust enough in practice because of heavy dependence on KLT tracker which is not robust in cluttered background and presence of occlusion. In [11] a hierarchical face tracking algorithm was proposed using one PTZ camera and one fixed camera. When the PTZ camera loses the face, their system locates it from the fixed camera. A simple background subtraction method was adopted for face tracking which limited the system from practical situations where the object moves freely in large areas and dynamic backgrounds. Mian [17] presented a real-time face detection and tracking

using a PTZ network camera. The location of the face was tracked using CAMSHIFT (Continuously Adaptive Mean Shift) [4] which relies on skin color of the face based on Hue channel from HSV color model. Although the method runs very fast, it faces the challenge of only locating the skin-color area which is easily distracted by other faces and skin-like color areas in the scene. Also using CAMSHIFT, Kumar *et al*. [16] extend the system into non-rigid object tracker, in which the color histogram is replaced by the probability for each pixel to belong to an object. Even using a different object representation, the method still fail by using color to differentiate between object and background, which is easily broken in complex environments. Going beyond tracking, Micheloni *et al*. [18] proposed an integrated framework where the event recognition and face recognition step were also taken account. One static and one PTZ camera were employed; the static camera was in charge of people detection and event recognition so that only the object involved in an anomalous event would trigger the active tracking from the PTZ camera. However, like [17], this method also used skin color-based CAMSHIFT tracker [4] and another fixation point tracker [21] to determine the center of the object, it cannot deal with cluttered background.

More recently, Dinh *et al*. [7] proposed a real-time object tracking system using an active PTZ network camera. The system contains a two-stage particle filter in which two stages were taken place by a color discriminative classifier and a generative model using single linear subspace, respectively. This method also adopted KLT tracker to cooperate with the generative model to locate the target. A time sharing strategy was proposed to adjust the priority of sending control commands in order to address the network delays and slow response when using PTZ camera over network. Also using color of the object and the background to learn a discriminative classifier, this system faces the same drawback like the proposed method in [16]. Given that the KLT tracker is very sensitive to occlusion also limits the system from reliably following the object after occlusion. Moreover, it is impractical for this method to reacquire the object after leaving the FOV, as exhaustive search over the whole input image is expensive for the proposed models. In summary, most existing systems fail by using a simple tracker due to strictly real-time requirement, which is vulnerable against challenging situations in practice. Moreover, most of them assume the face is visible with a reasonable resolution in the FOV at the beginning, so that the face detector is able to find the face of interest. This assumption is hardly valid in practice.

### *C. Overview of our approach*

To address all of the above issues, we present an autonomous system running in 3 different image modes, and 2 camera control modes. The overview is shown in Figure 2, where the operators acting on image and on camera are illustrated in light blue and light orange boxes, respectively. Our system uses a PTZ network camera to capture image sequences. The detector in pedestrian detection mode detects

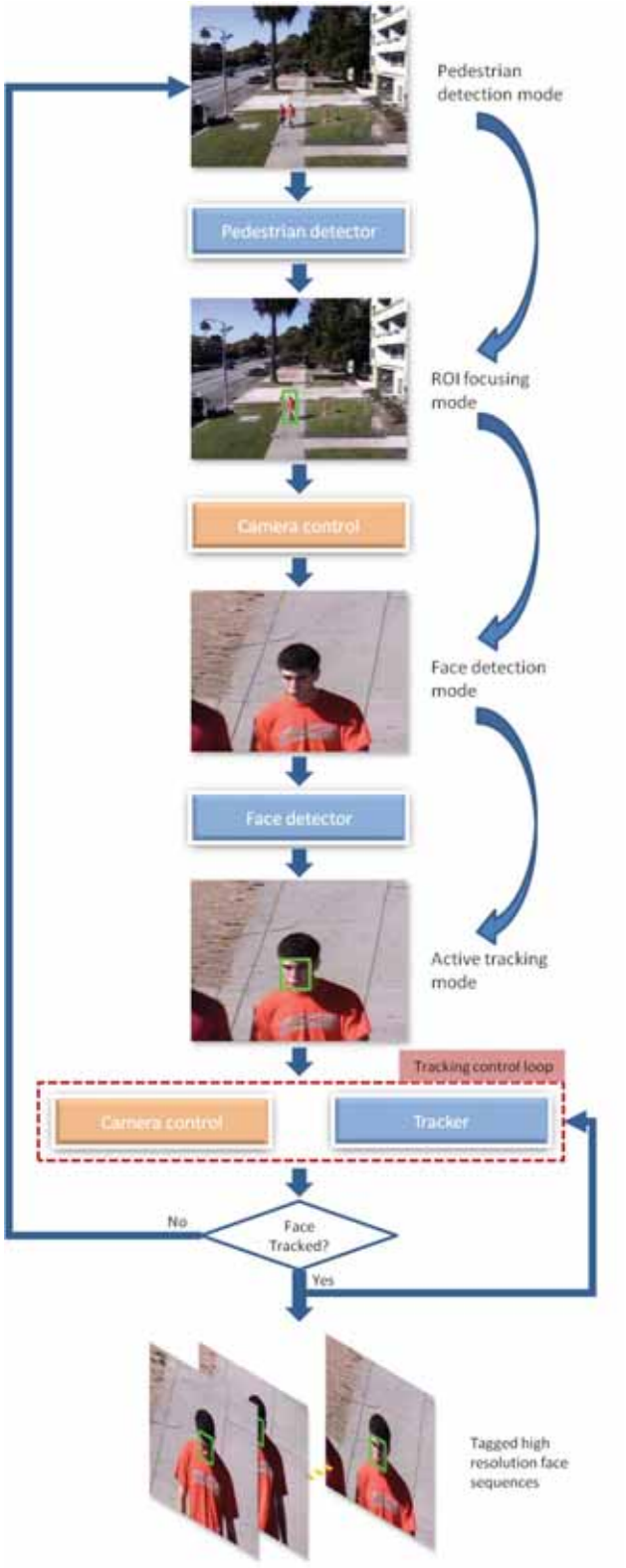


Fig. 2. Overview of our system

people in the FOV. The camera then switches to ROI focusing mode in order to zoom in the upper part of the detected

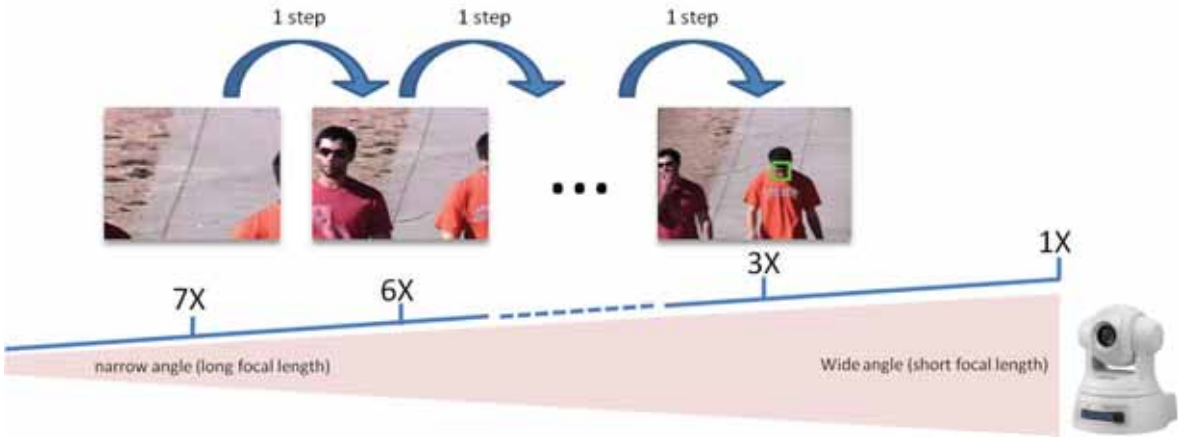


Fig. 3. One-step-back strategy camera control

human body. After that, the system goes into face detection mode to find the face of interest, which was hard to detect from far away in wide angle focal length (Figure 1). An active tracking mode consiting a control loop with two interchangeable modules: camera control and tracking ones is then triggered to follow the target by keeping it around the center of the image at a proper predefined size.

The rest of this paper is organized as follows. The components of our system are described in section II. The experiments are demonstrated in section III, followed by conclusions and future work.

## II. COMPONENTS OF OUR SYSTEM

### A. Pedestrian detection module

Many methods model the background then apply a background subtraction algorithm to detect moving objects. However, it is impractical to model the background under all of possible variations of PTZ camera parameters. Moreover, the environment conditions can change all of the time. In practice, there are not only pedestrians moving but also other objects such as vehicles and foliage in wind. Hence, we want to generalize the problem by employing a frame-based state-of-the-art pedestrian detector to find a walking person in the camera's FOV. Several pedestrian detection methods were proposed such as [6], [22]. However, most of them do not have near real-time performance. Recently, Huang *et al*. [13] proposed a high performance object detection using joint ranking of granules (JRoG) features. With a simple ground plane estimation, this detector takes only 70 ms to scan a 640x480 test image at 16 different scales. JRoG is a simplified version of APCF [8] descriptor after excluding gradient granules. For feature selection, two complementary modules which are an SA and an incremental module were successfully proposed to find the optimal solution. After that, the part-based detection approach [22] is adopted to find the optimal combination of multiple partitions of body parts in the complex scene. For more details please refer to [13]. To avoid false alarms, we use the simple frame difference background subtraction technique to filter out most of static region in the background. Also, it is important to note that we do not need to run the pedestrian detector in every frame, and we continuously run it once per second instead.

### B. ROI focusing module

After receiving the results from the pedestrian detector module, the system automatically chooses the highest confidence response and switches to the ROI focusing mode. Roughly, the head of a person is about $1/5$ the total height of whole body. In this mode, the camera parameters are adjusted so that the head position is close to the center of the image while its height is in the defined range. More clearly, let $C(c_x, c_y)$ denote the center of the image, $P(p_x, p_y)$ the center and $h_p$ the height of the ROI in the current state. The camera pans, tilts, and zooms so that on the image plane:

$$\begin{cases} P \rightarrow C \\ h_p \rightarrow min_h \quad if \quad h_p < min_h \\ h_p \rightarrow max_h \quad if \quad h_p > max_h \end{cases} \tag{1}$$

where $min_h = 80$ and $max_h = 120$ are the minimum and maximum heights of the face, respectively. Because this function only needs to be performed once, absolute commands are sent to control the camera. The output of this mode is a new video stream where the face in high resolution is likely in the image. It is important to note there is some possibility that no face exists in the new video stream: a false alarm from pedestrian detector, a person facing back or a fast moving pedestrian.

### C. Face detection module

This mode is in charge of detecting the face in the current image. We employ the real-time face detection proposed by Froba and Ernst [10]. This method introduces a new feature set, illumination invariant Local Structure Features for object detection. The new feature set are computed from a 3x3 pixel neighborhood using a modified version of Census Transform [24] for efficient computation. A cascade of classifiers of four stages is adopted, each of which stands a linear classifier consisting a set of lookup tables of feature

weights. The detection is then carried out by scanning all of possible windows with a fixed size of 22x22 pixels for each scaling factor of 1.25 until the size of the image is less than 22x22. Among the set of detected responses, the best one is chosen as the target. This face detector runs at 20fps on 640x480 image sequences and can detect faces with less than $45^\circ$ out-of-plane rotation. For more information please refer to the original work in [10].

Once the face is detected, the active tracking mode is triggered with two modules: camera control and tracking.

### D. Camera control module

This mode automatically controls the camera to follow the object. The control needs to perform smoothly and precisely. Following the efficient control framework proposed by Dinh *et al*. [7], we use relative control commands with a truncated negative feedback to avoid oscillation. When sending commands to the camera, we adopt a time sharing strategy in which one pan-tilt command and one zoom command are packed as one sending group, and two sending groups with one inquiry command make a control loop. This is because the camera queue is limited and we should not send commands frequently; otherwise, when the queue is full, later commands are dropped. Moreover, the delay in responding to the inquiry command is longer than the others, so a proper time sharing strategy is helpful. Note that most PTZ cameras carry the commands by mechanical processes which accumulate errors by time. This fact prevents us from acquiring the current status camera based on the initial status by computing the difference at every step.

In practice, it is hard to follow the face all of the time, especially when focal length is long, *i.e*. when the camera zooms in on the far away face. The FOV of the camera is narrow, and the face can move out of the image in 2-3 frames. To address this issue, we introduce a one-step-back strategy (as shown in Figure 3). We divide the zoom range in nine steps covering from 1X to 18X. We ignore the further zoom because the FOV is too narrow at that time. Whenever the face is no longer tracked after $k = 20$ consecutive frames, the system zooms out one step back with the hope to reacquire the face again. The system iteratively continues this process until it can re-acquire the face (using the tracker, not the detector), otherwise the camera gets back to the widest mode. The system comes back to the home position after missing the target after 10 seconds of no detection. The home position is preset.

### E. Tracking module

Together with the control component in the active tracking mode, the tracker in this mode is a critical component in our system. As discussed in Section I, the tracker is required to perform in real time and be able to flexibly adapt to appearance changes, condition changes, while being robust again abrupt motion, motion blur, cluttered background, etc. Moreover, the tracker also needs to reacquire the object when it leaves the FOV or reappears after total occlusion.

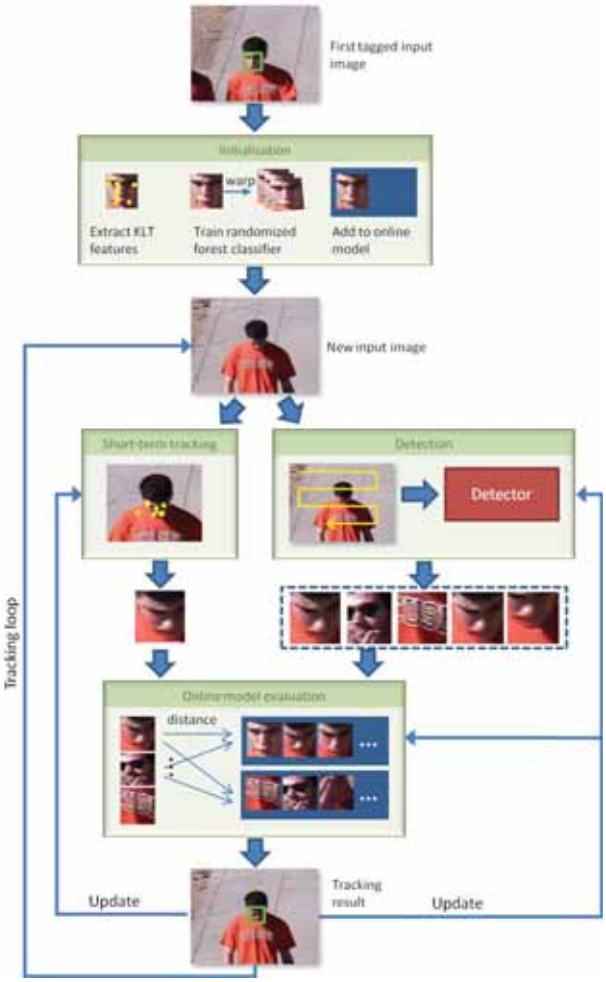


Fig. 4. Tracking flowchart

Recently, a large number of state-of-the-art tracking methods have been proposed with very good results [23], [1], [2], [12], [15]. Most of them assume that there is smooth motion between frames and fall into sampling based searches for efficient computation. However, this assumption is not guaranteed in practical scenarios where PTZ network cameras are used. The problem comes from the delays of network communication and command response of the camera, which lead to various challenging conditions including abrupt motion and motion blur.

Inspired by the tracker developed from P-N Learning strategy [15] (PNTracker) which was developed from the original online learning algorithm Tracking-Modeling-Detection (TMD) [14], we propose a new tracker by exploring different feature sets to choose the most suitable and robust feature for our system. Our tracker is also based on tracking-by-detection concept where scanning window is employed to exhaustively search for the object of interest. The P-N learning process is guided by two constraints which are positive (P) and negative (N) ones. These constraints help the tracker to explore incoming unlabeled data. There are

three components in PNTracker: a short-term Lucas Kanade tracker [20], an object detector using a randomized forest classifier [5], and a template-based object model which is online updated. The flowchart of the tracker is shown in Figure 4. It works as follows:

- Short-term tracking: The KLT features inside the object from the previous frame are extracted and tracked to the current frame. The object location is computed based on the median of the features.
- Detection: a scanning window technique is applied for exhaustive search in the current frame. Each sample is evaluated by the randomized forest classifier. All candidates passing the randomized forest classifier and the best sample extracted from a tracked region are transferred to the the object model for final decision.
- Online model evaluation: the distance to object model for all candidates are computed. The best candidate is chosen as the result at the current frame, while others which are far from the result location are used as negative samples. The positive samples are drawn randomly around the position of the result. Those positive and negative samples are used to train the randomized forest classifier and to update the online object model.

**Object detector**

*1) Features:* In PNTracker [15], the authors used 2-bit Haar-like Binary Pattern features (2bitBP) which measure gradient orientation in a randomly selected rectangular area, quantize it and output four possible codes which can be encoded in two bits. The gradient orientation is computed by comparing the sum of intensity values inside the areas split by horizontal and vertical lines in a selected rectangle. Here, to examine the efficiency of the feature sets to be used, we introduce other two feature sets. One of them is the 6bitBP which is extended from 2bitBP by applying pairwise relationship between the inside areas of a randomly chosen rectangle and itself, and between that rectangle with the whole image patch. The other, which is called DisjointBP, is simplified from the JRoG feature set [13], in which we increase the randomness of the algorithm by choosing random 4 same-sized rectangles to compute the feature values. For better clarity, we consider $f$ as a single feature in our feature set, so that a bit $i$ of feature $f$ is computed simply as follows

$$f_i = \begin{cases} 1 & if \quad Sum_i(r_w) > \alpha Sum_i(r_b) \\ 0 & otherwise \end{cases} \tag{2}$$

where $r_w$ and $r_b$ are the white and black rectangles, respectively and $\alpha$ is the ratio difference of the areas. In other words, the value of a feature $f$ is a concatenation of bits corresponding to the difference intensity of several pairs of non-overlapped rectangles. The illustration of feature sets is shown in Figure 5. Note that the DisjointBP feature also has 6-bit value. Detailed comparisons between these feature sets are given in Section III.

*2) Randomized forest classifier:* The forest contains a number of ferns [19] which can be considered as simplified trees. Each fern corresponds to a number of Binary Pattern features. A leaf in the fern records the number of positive $p$ and negative samples $n$ which are added during the training process. A sample is evaluated by calculating its feature values to find the corresponding leaf in the fern, and then the posterior is computed by maximum likelihood estimator $Pr(y = 1|x_i) = p/(p + n)$. If a sample cannot go to a leaf of a fern, its posterior is set to zero. The final posterior of the sample is the average number evaluated from all ferns. The forest filters out all samples whose final posterior less than 50%. Following [15], we also train the initial classifier in the first frame by generating 1000 positive samples from the first selected object by affine warping and add them to the forest. At every frame, all of the scanning samples are evaluated by this classifier. After having the result decided by the online object model, all samples having passed the classifier as positive but far from the result location are added to classifier as negative samples for bootstrapping.

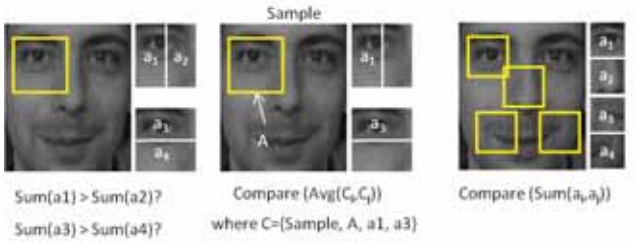


Fig. 5. Feature sets. From left to right: 2bitBP, 6bitBP, and DisjointBP

**Online object model**

In PNTracker [15], instead of adopting the online model, only the initial image patch in the very first frame was used which is a very conservative approach. It limits the tracker to adapt to appearance changes. Given that, we follow the online model idea proposed in [14] which provides more flexibility. Here, we extend the online model to store not only the positive samples but also the negative ones as well. The reason is to maximize not only the similarity between a sample to be checked and the positive samples, but also its discrimination against the negative ones. In details when a sample is checked by the online model, its distance to positive and negative image patches are calculated as $d_p$ and $d_n$, respectively. The distance of the sample to a model is then defined as $d = d_p/(d_p + d_n)$. It means that the optimal solution minimizes the distance between the checking sample and the positive set while maximizing the distance between it and the negative one. To calculate the distance between two image patches *Normalized Cross-Correlation (NCC)* is used:

$$d(x_i, x_j) = 1 - NCC(x_i, x_j) \tag{3}$$

To avoid biased decisions when existing several image patches in the online model are close to the testing sample, we calculate the average distance of $n_p$ and $n_n$ shortest ones instead of using only the closest image patch from each positive and negative set. In our implementation, we choose $n_p = 3$ and $n_n = 1$.

If distance of tracked object is larger than a defined threshold $\theta_{lost} = 0.5$, the tracker considers the object be

missing in the FOV. Whenever the best distance result in the current testing goes to a a smaller value than a defined $\theta_{track}$ threshold which is set $\theta_{track} = 0.35$, the tracker triggers the state back to normal.

Moreover, the trajectory is validated at every frame to guarantee that the right target is tracked. This validation step is similar to the growing event proposed in [14] and basically relies on two selection strategies: Difference Between Consecutive Patches (DIFF) and Temporal Pattern (LOOP). When the distance of the last sample in trajectory is smaller than a defined threshold $\theta_{track}$ (discussed above), or the distances between consecutive samples in trajectory is smaller than a defined threshold $\theta_{step} = 0.2$. This step allows the model to accept validated trajectory with adaptive or loop patterns, thus, enables the flexibility of object modeling.

However, to avoid the ineffective increase of image patches in online object model by adding samples which are similar to existing ones. We only consider to add any sample with distance to the model larger than a defined threshold $\theta_{addpositive} = 0.1$. Also, it reflects the idea of building a multiple-view model of an object. The negative samples are added to online model in the same way with $\theta_{addnegative} = 0.05$.

It is important to note that when checking a sample there are often several similar existing poses in the object model. Therefore, instead of choosing only the closest one to compute, we take the average distance to $n_p = 3$ image patches with shortest distances in the model.

## III. Experiments

### A. Tracking validation

To validate our tracker, we compare it with several state-of-the-art methods in terms of precision error and running time. The experiments are carried on a system Intel Xeon 3.00Ghz with a single thread. The trackers included in this comparison are FragTracker (FT) [1], Co-Training Tracker (CoTT) [23], P-N Learning Tracker (PNT) [15] and our tracker with different feature sets which are 2bitBP, 6bitBP, and DisjointBP. Note that we use 6 ferns in our implementation while Kalal *et al*. [15] reported to use 10 ferns. All of the implementations are in C++, except PNT was implemented in C-Mex. The testing data set includes seven image sequences with different resolutions: Scale and Clutter from ETH Zurich computer vision lab evaluation data set[1], Girl from [2], Multiple faces from [14], Vehicle, Jumping, and Multiple persons from [23]. These sequences containing different types of objects (faces, vehicle, pedestrian, toys) in different environments: indoor (Scale, Clutter, Girl, Faces) and outdoor (Vehicle, Jumping, Person). The challenging conditions include cluttered background (Clutter, Faces, Vehicle, Person, Girl), motion blur (Jumping, Clutter), leaving FOV or total occlusion (Girl, Faces, Vehicle, Person), abrupt motion or abrupt appearance changes (Scale, Clutter), large changes in pose (Scale, Girl, Faces, Vehicle).

[1]Tracking evaluation: http://www.vision.ee.ethz.ch/boostingTrackers/

| Res | FT | CoTT | PNT | 2bit | 6bit | Disjoint |
|---|---|---|---|---|---|---|
| 320x240 | 1.8 | 4.5 | 13.6 | **34.2** | 31.5 | 29.2 |
| 352x288 | 1.6 | 4.2 | 7.9 | 22.8 | **23.8** | 22.1 |
| 640x480 | 0.7 | 3.9 | 3.7 | 14.3 | **17.1** | 12.5 |
| 720x576 | 0.5 | 3.1 | 3.4 | 12.8 | **15.6** | 11.1 |

TABLE II

AVERAGE RUNNING TIME IN FRAME PER SECOND (FPS). (FT: FRAG-TRACKER [1], COTT: COTRAINING-TRACKER [23], PNT: P-N TRACKER [15]) IN DIFFERENT RESOLUTION. THE BEST PERFORMANCE IS IN BOLD.

Table I shows the comparison results between the methods. The average center location error (in pixels) is used as the measurement as in [2]. We prefer to use this metric instead of the detection-criterion of the VOC Challenge [9] because different methods have different ways to initialize their tracker. Some of them prefer a loose bounding box while the others prefer a tight one. We also show the running time comparison in Table II. Note that except from the PNTracker [15] and ours, all others define a search range to find their best samples. The larger the search range, the slower the method. We chose a reasonable search range for each of them in a trade-off with running time: CoTT [23] (60 pixels), FT [1] (30 pixels). The CoTT increases the search range up to the maximum of 120 pixels when it cannot find the object in the current image. Because all other trackers show not much different results, for the sake of clarity, we show some comparison snapshots between our tracker using 6bitBP feature and FT [1] in Figure 6. The results prove that our three trackers corresponding to three different data sets perform robustly on all of the sequences while the running time is very fast compared to other state-of-the-art methods. Due to the similar tracking quality, we choose the fastest tracker, the 6bitBP, for the tracking module in our system.

### B. System validation

*1) Experimental setup:* We use the computer system described in section A with a Sony PTZ network camera SNC-RZ30N. This camera covers a large range of pan angles ($-170°\rightarrow +170°$ ), tilt angles ($-90°\rightarrow +25°$), and a large zoom ratio ($25X$ optical). The camera image stream sent to the computer is set at 640x480 with medium quality. All parameters are fixed as described in the other sections. Our complete system runs at 15fps. The complete system has been tested in challenging indoor and outdoor environments with real-life situations. Our system successfully detects a walking person, zooms in on his face and keeps track for a while. For indoor (see Figure 7), it is very challenging due to the face leaving FOV, large viewpoint changes, lighting condition changes. For outdoor environment, we tackle two different situations: one is in the afternoon with enough lighting, one is in the evening with low light. The results show that even when the camera cannot focus on the face with auto-focusing mode, and there is a lot of noise, our system still delivers acceptable results. Some snapshots are shown in Figure 8. It is important to note that, to reduce the

| Video Sequence | Frames | Resolution | FT | CoTT | PNT | 2bit | 6bit | Disjoint |
|---|---|---|---|---|---|---|---|---|
| Scale | 1911 | 640x480 | 9 | **5** | 9 | 6 | 6 | 6 |
| Clutter | 1528 | 640x480 | 92 | **4** | 7 | 7 | 11 | 9 |
| Girl | 502 | 320x240 | 70 | 14 | 25 | 16 | **14** | 16 |
| Faces | 1006 | 720x576 | 159 | 111 | 8 | 40 | **26** | 35 |
| Vehicle | 945 | 320x240 | 59 | 9 | **6** | **6** | **6** | 7 |
| Jumping | 313 | 352x288 | 10 | **3** | 8 | 12 | 12 | 12 |
| Person | 338 | 320x240 | 81 | 26 | 31 | 14 | 14 | **11** |

TABLE I

AVERAGE CENTER LOCATION ERROR. (FT: FRAG-TRACKER [1], COTT: COTRAINING-TRACKER [23], PNT: P-N TRACKER [15]) IN DIFFERENT CHALLENGING DATA SETS. THE BEST PERFORMANCE IS IN BOLD.

(a) Clutter sequence at frame 20, 140, 615, 900, 1369, 1525 from left to right

(b) Faces sequence at frame 35, 105, 255, 615, 783, 930 from left to right

(c) Vehicle sequence at frame 80, 300, 390, 570, 730, 920 from left to right

Fig. 6. Comparison between our 6bitBP tracker (in solid green) and FragTracker(FT) [1] (in dashed pink)

Fig. 7. Indoor experiment results

(a) Outdoor environment with enough lighting

(b) Outdoor environment with low-light and noisy, out-of-focus images

Fig. 8. Outdoor experiment results

(a) Scale sequence at frame 250, 975, 1500 from left to right

(b) Girl sequence at frame 30, 330, 470 from left to right

(c) Jumping sequence at frame 100, 220, 313 from left to right

(d) Person sequence at frame 30, 165, 320 from left to right

Fig. 9. Some snapshots from the results of our 6bitBP tracker

overhead of recording images, we only save the sequences of images after the face is detected and starts to be tracked. For detailed results, please refer to our supplemental video. Some other results of our 6bitBP tracker on other image sequences are shown in Figure 9.

## IV. CONCLUSION AND FUTURE WORK

We have presented a novel real-time system to acquire high resolution sequences of a person's face using a PTZ network camera. Our proposed system is fully autonomous and operates in four different robust modes: pedestrian detection, ROI focusing, face detection and active tracking. Our system addressed most of the issues efficiently in the control module such as zooming control, network delay, slow command response and in the tracking module such as appearance changes, motion blur, cluttered background and reacquisition. We have validated our approach on a variety of real outdoor scenarios. Our future research will address sequencing strategy when multiple people are detected, interfacing to face recognition module, and possibly to a 3-D modeling from video.

## V. ACKNOWLEDGEMENTS

This research was funded, in part, by MURI-ARO W911NF-06-1-0094. The first author was also supported by the Vietnam Education Foundation. We thank Eunyoung Kim and Quang Vu for their help in collecting the data. We also thank to Zdenek Kalal and Dr. Jongmoo Choi for helpful discussions.

## REFERENCES

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798–805, 2006.
[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, pages 983–990, 2009.
[3] A. D. Bagdanov, A. del Bimbo, and W. Nunziati. Improving evidential quality of surveillance imagery through active face tracking. In *ICPR*, volume 3, pages 1200–1203, 2006.
[4] G. R. Bradski. Computer video face tracking for use in a perceptual user interface. In *Intel Technology Journal*, 1998.
[5] L. Breiman. Random forests. In *ML*, volume 45, pages 5–32, 2001.
[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893, 2005.
[7] T. Dinh, Q. Yu, and G. Medioni. Real time tracking using an active pan-tilt-zoom network camera. In *IROS*, pages 3786–3793, 2009.
[8] G. Duan, C. Huang, H. Ai, and S. Lao. Boosting associated pairing comparison features for pedestrian detection. In *The Ninth IEEE International Workshop on Visual Surveillance*, pages 1097–1104, 2009.
[9] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge. In *VOC*, 2007.
[10] B. Froba and A. Ernst. Face detection with the modified census transform. In *AFGR*, pages 91–96, 2004.
[11] T. Funahasahi, M. Tominaga, T. Fujiwara, and H. Koshimizu. Hierarchical face tracking by using ptz camera. In *FGR*, pages 427–432, 2004.
[12] H. Grabner, C. Leistner, and H. Bishof. Semi-supervised on-line boosting for robust tracking. In *2008*, pages 234–247, ECCV.
[13] C. Huang and R. Nevatia. High performance object detection by collaborative learning of joint ranking of granules features. In *CVPR*, pages 41–48, 2010.
[14] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *OLCV*, pages 1417–1424, 2009.
[15] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, pages 49–56, 2010.
[16] P. Kumar, A. Dick, and T. S. Sheng. Real time target tracking with pan tilt zoom camera. In *DICTA*, pages 492–497, 2009.
[17] A. Mian. Realtime face detection and tracking using a single pan, tilt, zoom camera. In *IVCNZ*, pages 1–6, 2008.
[18] C. Micheloni, E. Salvador, F. Bigaran, and G. L. Foresti. An integrated surveillance system for outdoor security. In *AVSS*, pages 480–485, 2005.
[19] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *CVPR*, pages 1–8, 2007.
[20] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.
[21] B. J. Tordoff and D. W. Murray. Reactive control of zoom while tracking using perspective and affine cameras. In *PAMI*, volume 26, pages 98–112, 2004.
[22] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. In *IJCV*, volume 75, pages 247–266, 2007.
[23] Q. Yu, T. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers and discriminative trackers. In *ECCV*, 2008.
[24] R. Zabih and J. Woodfill. A non-parametric approach to visual correspondence. In *PAMI*, 1996.