

# Realtime Face Detection and Tracking Using a Single Pan, Tilt, Zoom Camera

Ajmal Mian

School of Computer Science and Software Engineering  
The University of Western Australia  
35 Stirling Highway, Crawley, WA 6009, Australia  
Email: [ajmal@csse.uwa.edu.au](mailto:ajmal@csse.uwa.edu.au)

## Abstract

*Surveillance cameras generally have a wide field of view and do not capture the human face with a resolution that is sufficient for machine recognition. To overcome this problem, this paper presents a realtime face detection and tracking algorithm using a single PTZ (Pan, Tilt, Zoom) camera. Unlike existing algorithms which track the human face in the field of view of a fixed camera, the proposed algorithm adjusts the pan, tilt and zoom of a PTZ camera in real-time so that human faces can be acquired at the camera's maximum resolution. The proposed algorithm addresses challenges like network delays, camera movement lags, oscillations and lost faces. Experiments were performed in realistic surveillance scenarios and accurate real-time tracking with pan, tilt and zoom was achieved.*

**Keywords:** Face tracking, PTZ camera, smart surveillance.

## 1 Introduction

The human face is one of the most attractive biometrics as it can be acquired without the cooperation or knowledge of the subject. However, machine recognition of faces is a challenging problem as the appearance of a face significantly changes with illumination, pose and facial expressions. Moreover, facial biometrics are comparatively less distinctive [1]. A comprehensive survey of face recognition algorithms is provided by Zhao et al. [2] however, a brief survey is presented here to put this paper into context.

Face recognition from still images has been investigated for a long time. One of the earliest algorithms in this category is the eigenfaces [3] which projects the facial images to PCA subspace and performs matching on a subset of the most significant eigenvectors. This approach is simple however, it requires perfect normalization of the faces and is sensitive to variation in illumination, pose and expressions. Fisherfaces [4] perform Linear Discriminant Analysis (LDA) for face recognition and requires multiple instances per face e.g. under varying illumination, pose or expressions. The idea is to project the faces such that intra-class variation is minimized and inter-class variation is maximized. It assumes that the classes are linearly separable which is not always true in the case of

faces. Moreover, intra-class variation due to changing illumination and pose can cause more variation in faces compared to inter-class variations. Methods like eigenfaces and fisherfaces which extract features from the face as a whole are called holistic methods [2]. On the other hand, algorithms which extract local features, such as the eyes and nose, and match their relative locations or statistics are called feature-based methods [2]. An examples of feature-based methods is the elastic-bunch graph matching [5]. Compared to holistic methods, feature-based methods require higher resolution facial images to extract good quality local features. Face recognition algorithms that extract both holistic features and local features are called hybrid methods [2].

Compared to still images, videos contain more information because of the additional temporal dimension. Motion also helps in the recognition of faces [2]. This makes video-based face recognition one of the most viable options for surveillance applications. However, surveillance is a very challenging scenario for face recognition. Surveillance cameras have a wide field of view because they are required to survey large areas. Moreover, surveillance cameras are also meant to monitor activities rather than just focus on faces. However, the downside of a wide field of view is that only a limited number of the pixels cover the human face which is insufficient for machine recognition.

In the past few years, there has been an influx of PTZ (Pan, Tilt, Zoom) cameras in the market which has made it feasible to track and zoom in on human faces. PTZ cameras are more suitable for surveillance because of three reasons. One, they can pan to cover the entire 360 degrees field of view and tilt to adjust the viewing angle. Two, they also have the advantage of zooming out to survey large areas (like conventional surveillance cameras) for activity monitoring or initial face detection. Three, they can zoom in on objects of interest e.g. human faces, in order to capture them at higher resolution.

Funahasahi et al. [6] proposed a hierarchical face tracking algorithm using a PTZ camera however, they used an additional fixed camera to detect the face. The additional camera simplifies tracking by updating the location of the face while the other camera is moving. If the PTZ camera loses the face e.g. at high zoom when its field of view is narrow, it can get the location of the face from the fixed camera. Amnuaykanjanasin et al. [7] also proposed face tracking with two cooperative PTZ cameras, one in static mode and the other performing pan and tilt. They did not perform zoom operation and performed prior calibration of the cameras. Removing zoom operation and using two cameras simplifies the face tracking to a great extent. Cindy et al. [8] performed vehicle tracking using a fixed and PTZ camera. Senior et al. [9] also used multiple cameras, fixed and PTZ, to get multiscale images of objects of interest. Although Tsotsos et al. [10] used a single PTZ camera, their aim was to define minimal camera parameters to survey the scene and segment the target using prerecorded images of the background. Our survey shows that existing research has mainly focused on PTZ tracking with multiple cameras and PTZ tracking with a single camera has received negligible attention.

This paper presents a real-time algorithm for face detection and tracking using a *single* PTZ camera. All three operations, i.e. pan, tilt and zoom, are simultaneously performed which is quite challenging because the field of view of the camera decreases with zoom and it is easy to lose the face. One of the major challenges in developing tracking algorithms to control PTZ cameras is that the algorithms cannot be trained or tuned offline after the data has been acquired as is the case with other computer vision algorithms that use fixed cameras. This is because the field of view and hence the output of PTZ cameras change with pan, tilt and zoom operations. Another major challenge is the delay caused by the servo motors after a movement command has been sent to the camera. Finally, network delays can vary the rate at which video frames arrive.

## 2 Video Acquisition

The Sony Ipela SNC-RZ50P network camera was used for experiments. Network cameras use the existing network infrastructure (intranets and the Internet) for video transfer as well as control commands for camera pan, tilt and zoom. Modern buildings already have a network in place therefore, the use of network cameras is cost effective. Moreover, cameras can be accessed from anywhere on the local network or even remotely via the Internet.

The camera accepts CGI (Common Gateway Interface) commands for requesting motion video, still images, setting and inquiry of camera parameters and control commands for panning, tilting, zooming and focusing the camera. The C++ Socket class was used to send CGI commands to the camera and retrieve motion JPEG video. Each frame was separated by detecting the string “myboundary” and written to a JPEG file in binary format. The frames can also be combined and written as an AVI file to the hard drive with little modification. However, the advantage of writing the frames to individual JPEG image files is that they can be easily previewed with the operating system using thumbnails.

## 3 Face Detection and Tracking

OpenCV [11] provides a set of open source programming libraries for low level and high level computer vision applications. To use OpenCV libraries, images must be loaded into the `IplImage` data structure. The JPEG image file formed in Section 2 was loaded into the `IplImage` data structure using `cvLoadImage`. Since images are read continuously in a loop, a memory leak can occur easily if memory resources are not overwritten or released at the end of the loop. Memory leaks, sooner or later, saturate the memory and the program crashes. Even though the JPEG image was loaded into the same `IplImage` data structure using the `cvLoadImage` function, it resulted in a memory leak proportional to the image size in bytes. This problem can be addressed by initializing `IplImage` and releasing it inside the loop using `cvReleaseImage`.

The OpenCV built in object detector which is based on Haar-like features was used to detect faces [12]. This algorithm [12] can detect faces in realtime and is thus suitable for our application. Haar-like features consist of simple black and white rectangles joined horizontally, vertically or rotated by 45°. The feature's value is the weighted sum of the pixels in the black and white rectangles. The weights of black and white rectangles have opposite polarity and are inversely proportional to their occupied areas.

Viola and Jones [12] proposed an elegant way of computing Haar-like features using an integral image. An integral image is formed by replacing each pixel by the sum of all pixels to its left and above. A search window is slid over the integral image and inside each window, Haar-like features are extracted using only a few sum and difference operations as opposed to a large loop. For scale invariant detection, the feature scale is varied rather than the image scale. This is much more efficient as the integral image has to be calculated only once.

Multiple weak classifiers are trained using a labeled set of negative and positive face images. These weak classifiers are then cascaded to form a complex strong classifier using AdaBoost [13]. OpenCV ships with already trained classifiers for frontal and profile face detection. We used the frontal face detection in our experiments. The image was converted to grayscale and histogram equalized before face detection. Since the rest of the procedures (tracking and camera movement) rely on accurate initial face detection, we only considered faces that were detected at multiple different resolutions.

The location of the detected face was then passed on to CAMSHIFT (Continuously Adaptive Mean Shift) [14] face tracker of OpenCV. CAMSHIFT creates a probability distribution of the facial skin color in HSV (Hue Saturation Value) images. The HSV model is suitable for tracking because it separates color from saturation and brightness making the model less sensitive to illumination. A 1D histogram is constructed from the hue channel of the already detected face at run time and stored as a model. Next, the pixels of the incoming frames are converted to a corresponding probability of facial skin which is tracked in the  $x, y, z$  and roll space ( $x, y$  are pixel values,  $z$  is the scale which depends upon the distance of the face from the camera and roll is the head roll along the  $z$ -axis). Note that the face detection algorithm [12] is trained to detect front views of faces but once a face has been detected, the tracker [14] can track it in any pose. Another advantage of CAMSHIFT is that it operates in realtime and utilizes minimal CPU resources leaving out plenty for other operations like face recognition.

## 4 PTZ Tracking Algorithm

Three of the four output tracking parameters of the CAMSHIFT namely the  $x, y$  and  $z$  were used to update the pan, tilt and zoom of the PTZ camera respectively. Moreover, autofocus function of the camera was used. The roll parameter was not used for camera movement as PTZ cameras do not have roll capability. The roll information could be used for face normalization however, we did not use it as more accurate face normalization techniques exist in the literature.

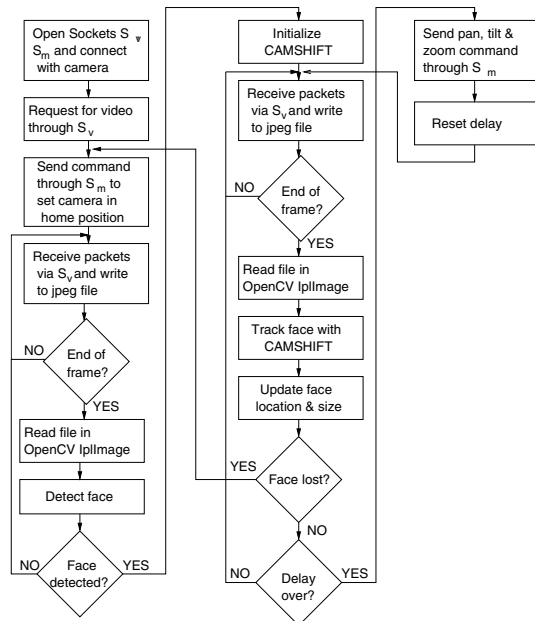
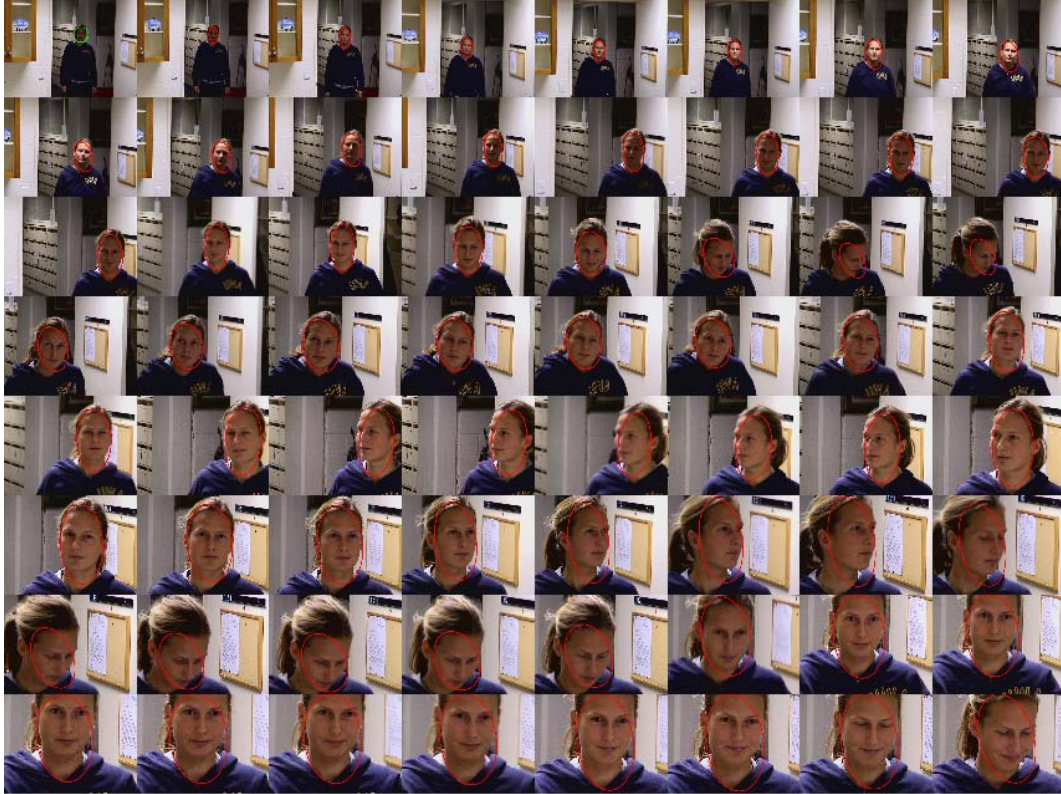


Figure 1: Block diagram of the tracking algorithm.

Recall that JPEG video is streamed from the PTZ camera via a TCP (Transport Control Protocol) connection using sockets. Sending control commands to the camera via the same socket would introduce unnecessary video delays and complexity in terms of differentiating between incoming video packets from movement command execution confirmation packets. Therefore, a separate TCP socket was created for sending control commands to the PTZ camera. The Sony SNC-RZ50P provides two methods for sending control commands. In the first method, the socket is kept open by setting the “KeepAlive” parameter and control commands are sent using the same socket. Apart from executing the movement commands, the camera also responds to this method with a header containing no data. This header must be read to free the buffers. The second method is more simple as the camera does not send any response packets and only executes the movement commands. However, in this case the connection is closed by the camera after receiving each command. The control software must also close the socket and re-open a fresh one to send the next command. Both approaches were tested to compare the response time of the camera and no difference was noticed because the camera movement delay was more significant compared to the socket creation delay. We created a separate socket (in addition to the video socket) once and used it through out the run time to issue control commands to the camera. However, the incoming confirmation packets were not read (to free up the buffer) after every control command but once every  $N$  number of frames to save time.

Video arrives at 25 frames/sec. and the location of face may change from frame to frame. However, most commercial PTZ cameras cannot change their



**Figure 2:** Video frames of a tracking sequence (every 5th frame is shown). The face is detected (1st frame) and then tracked. The camera adjusts its pan and tilt and continuously zooms to capture the face at its full resolution.

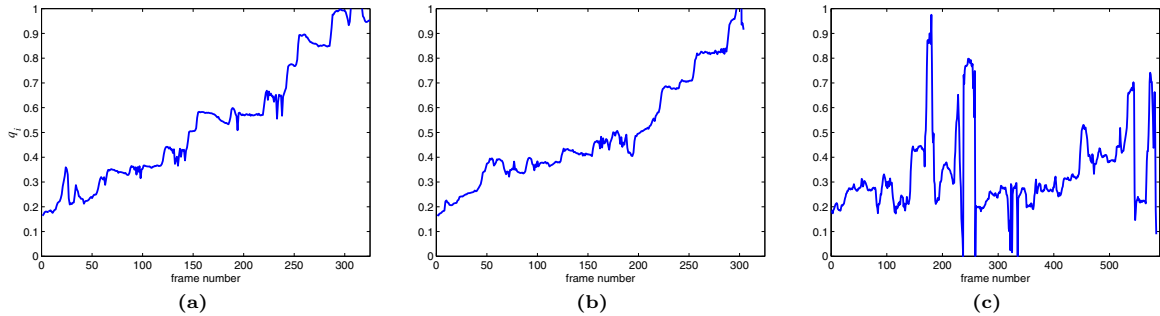
orientation at this speed. It was observed that the camera exhibits a significant delay while executing movement commands. This delay was measured to be 0.95 sec. for pan, tilt and another 0.85 sec. for 1.2 times zoom operation. The camera first performs pan and tilt followed by zoom operation. The former at a higher speed. The speed of the camera puts an upper limit on the tracking. To ensure that this does not affect the within frame face tracking, the face was tracked from frame to frame and a delay was introduced after every movement command giving the camera enough time to change its orientation. Without this delay, the camera goes into oscillations. The delay was calculated from the CPU clock as opposed to the frame rate which can vary due to network traffic. Moreover, zooming was limited to approx. 1.2 at a time so that the face does not go outside the field of view of the camera due to sudden movement of the subject while the camera is zooming. The camera iteratively changes its orientation and zoom to capture the face at its highest resolution.

Due to the camera movement delay, it is possible that a face is lost due to quick movement of the subject. This is more likely to occur at high zoom when the camera's field of view is very narrow. When a face is lost during tracking, the  $z$  parameter (width) of CAMSHIFT becomes very small. Under these circumstances, the tracker was stopped, the camera zoomed out and face detection

was re-initiated. The camera can either return to its home position to detect and track the next face or zoom out at the same orientation to track the same face again. Fig. 1 shows a block diagram of the proposed PTZ face tracking algorithm.

## 5 Experimental Evaluation

Experimental evaluation of the tracking algorithm was performed in two scenarios. In the first scenario, the subjects approach a front desk. This is a simulation of someone approaching an ATM machine or an immigration officer. The camera detects the face as it appears in its field of view and then tracks it continuously while changing its pan, tilt and zoom to acquire the face at full resolution. A fixed camera in this scenario will either miss the face if it is zoomed in (narrow field of view) or acquire the face at a low resolution if it is zoomed out (wide field of view). Fig. 2 shows an example tracking sequence. The tracking algorithm detects the face in the first frame and then tracks it by changing the pan and tilt of the camera so that the face is always centered in the frame. It also increases the zoom of the camera by a factor of approximately 1.2 each time until the height of the face approximately equals the frame height. Note that with every zoom operation, the spatial resolution of the captured face increases by the square of the increase in its height i.e. the resolution doubles after every two iterations of zoom.



**Figure 3:** Plots of  $q_i$  versus frame number in scenario one. (a) and (b) A face is detected, tracked and zoomed in until it reaches the frame height. (c) A challenging case where the subject abruptly appears and disappears (seven times) in the camera’s field of view at different distances. The algorithm performs good by detecting and tracking the face as it appears. The slumps and peaks represent lost and found faces respectively.

In the second scenario, subjects walk through a corridor. This is a simulation of a person walking through the hall way of an airport or the entrance of a shopping mall. In this case, the camera is situated far from the the subject which reduces its pan and tilt requirements compared to the previous scenario. However, not only more zoom operations are required but the camera must first zoom in on the subject and then start zooming out as the subject walks towards the camera for a longer period.

Currently, there is no criteria defined for the evaluation of PTZ tracking algorithms. Therefore, as our evaluation criterion, we plot the ratio of the tracked face height (in pixels) to the frame height  $h$  as a function of frame number i.e.  $q_i = f_i/h$  (where  $i = 0, 1, 2 \dots n$  is the frame number,  $f_i$  is the height of the face in frame  $i$ ). Fig. 3 shows the  $q_i$  plots for different subjects in scenario one. Higher slopes mean more efficient tracking but this is limited by the pan, tilt and zoom speed of the camera. On the bright side, the plots also tell us that if a subject takes at least 10 seconds at the front desk (including the time to approach it), his face will be acquired at the camera’s maximum resolution. A challenging case is also shown in Fig. 3-c where the subject tries to dodge the camera. Even though this is an unrealistic case, the algorithm performs quite well by acquiring a significant number of frames with  $q_i > 0.5$  (some at  $q_i = 0.9$ ).

Fig. 4 shows a sample tracking sequence and Fig. 5 shows the  $q_i$  plots for different sequences in scenario two. Since the camera is located far, zooming (per iteration) was increased to 1.6. Positive slopes represent zoom in operations and negative peaks represent zoom out operations in Fig. 5-a and 5-b. In Fig. 5-c, the subject tries to dodge the camera by walking forwards and backwards (facing opposite side) towards the camera. The negative and positive peaks in this case represent lost and found faces. The proposed algorithm was implemented in VC++ on a laptop with 1.66 GHz Centrino Duo processor and 1 GB RAM. It utilized 22% CPU and 11.6 MB memory resources. Source code and

sample tracking videos are available at [15].

## 6 Limitations

One obvious limitation of PTZ tracking is that a single face can be tracked at a time. This is because a single camera can center and zoom in on a single face at once making other faces, if any, outside its field of view. This limitation can be overcome to some extent in less crowded places. When sufficient number of frames of a subject’s face have been captured at the camera’s maximum resolution, the camera can then zoom out and track the next person. However, in crowded places, a single camera may not cope with sudden influx of people. Faster cameras or more number of cameras are required in such cases. Another major limitation of the proposed algorithm comes as a consequence of the limited pan, tilt and zoom speed of the camera. The former two are more significant limiting factors at close ranges and the last one is more significant at long ranges.

## 7 Conclusion

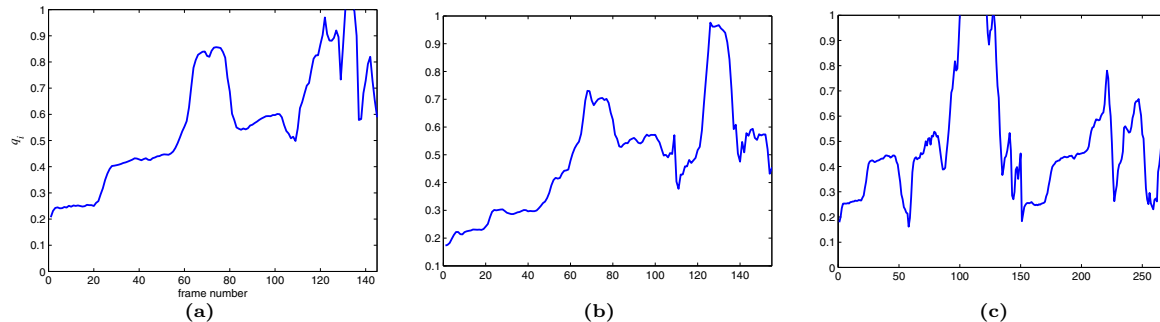
This paper presented a realtime face tracking algorithm using a PTZ camera. Unlike conventional algorithms, tracking with changing orientation and zoom of a camera is more challenging as a number of additional factors must be dealt with including, the movement delay of the camera and lost faces due to narrow field of view. To add to the complexity, PTZ tracking algorithms can only be tested and tuned online and not on prerecorded videos. Experiments were performed in realistic scenarios where the subjects were required to approach a front desk and walk through a corridor. Results show that the proposed algorithm can track faces and capture higher resolution images thereby facilitating subsequent machine recognition of faces.

## 8 Acknowledgments

Thanks to Joe Sandon for installing the camera. This research is funded by ARC grant DP0881813 and UWA Research Grant 2007.



**Figure 4:** Tracking sequence for scenario 2 (every 5th frame is shown). Last row: the camera zooms out when  $q_i$  crosses a threshold and then zooms in again.



**Figure 5:** Plots of  $q_i$  for scenario two. (a) and (b) A face is detected, tracked and zoomed in until  $q_i$  crosses a threshold after which the algorithm zooms out (represented by slumps) and starts tracking the face and zooming in again (second peaks). (c) A challenging case where the subject walks towards the camera sometimes facing backwards and sometimes facing forwards. The algorithm detects and tracks the face when it appears (peaks).

## References

- [1] A. K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," *IEEE TCSV*, vol. 14, no. 1, pp. 4–20, 2004.
- [2] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Computing Survey*, vol. 35, no. 4, pp. 399–458, 2003.
- [3] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, pp. 71–86, 1991.
- [4] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE TPAMI*, vol. 19, pp. 711–720, 1997.
- [5] L. Wiskott, J. Fellous, N. Kruger, and C. Malsburg, "Face recognition by elastic bunch graph matching," *IEEE TPAMI*, vol. 19, no. 7, pp. 775–779, 1997.
- [6] T. Funahasahi, M. Tominaga, T. Fujiwara, and H. Koshimizu, "Hierarchical face tracking by using PTZ camera," in *IEEE FG*, 2004, pp. 427–432.
- [7] P. Amnuaykanjanasin, S. Aramvith, and T. Chalidabhongse, "Face Tracking Using Two Cooperative Static and Moving Cameras," in *IEEE ICME*, 2005, pp. 1158–1161.
- [8] X. Cindy, F. Collange, F. Jurie, and P. Martinet, "Object tracking with a pan-tilt-zoom camera: application to car driving assistance," in *IEEE ICRA*, 2001, pp. 1653–1658.
- [9] A. Senior, A. Hampapur, and M. Lu, "Acquiring multi-scale images by pan-tilt-zoom control and automatic multi-camera calibration," in *WACV-MOTION*, 2005, pp. 433–438.
- [10] J. Tsotsos, K. Bennet, and E. Harley, "Tracking a person with pre-recorded image database and a pan, tilt and zoom camera," in *IEEE Work. on Visual Surveill.*, 1998, pp. 10–17.
- [11] Intel, "Open computer vision library," [www.intel.com/technology/computing/opencv/](http://www.intel.com/technology/computing/opencv/), 2008.
- [12] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.
- [13] Y. Freund and R. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Learning Theory*, pp. 23–37, 1995.
- [14] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, 1998.
- [15] A. Mian, "Tracking demo videos and source code," [www.cs.uwa.edu.au/~ajmal/](http://www.cs.uwa.edu.au/~ajmal/), 2008.