Control of a Pan-Tilt-Zoom (PTZ) Camera for Long Range Iris Acquisition of Tracked Subjects

Nick Vandal (nvandal@cmu.edu)

***Abstract*— Traditional biometric capture systems have a difficult time acquiring high quality iris images from a distance in an unconstrained environment. Actuating the large lens and camera assembly accurately to track a small moving object such an iris at long range (>10 meters) is challenging due to several factors including that long length of the lens assembly, the weight of the lens assembly, the presence of noise in target (face/iris) localization, the presence of noise in actuator feedback sensors, atmospheric effects, uncertainty in the distance to target, target movement, and sensor movement. We look at several methods of PTZ camera control and demonstrate an OpenGL based 3D PTZ control simulation.**

I. Introduction

Secure identification and authentication will obviously continue to be of vital importance in the future. Biometrics, which are unique physical characteristics of a person, offer many advantages over traditional means of authentication such as passwords and ID cards. This is primarily because they remain with the authorized person at all times and are difficult, if not impossible, to forge. The iris pattern is considered to be amongst the most reliable for high confidence identification due to the enormous variability among even genetically identical eyes [2][3]. Traditional biometric capture systems have a difficult time acquiring high quality iris images from a distance in an unconstrained environment and typically assume a close-range, cooperative subject or a very constrained environment with a natural funneling affect such as a narrow corridor. Attempting to achieve high resolution images of a small object such as an iris at a distance while being capable of wide area observations are two opposing goals, which necessitate requires high quality optics and cameras, as well as a control system capable of steering a camera with a narrow field of view. We seek to construct a system capable acquiring iris images with large standoff distances and with an unconstrained open environment, such as a field or city square.

A system shown in Fig. 1 with a powerful lens system mounted on a Pan-Tilt actuator should have the flexibility to achieve this goal. However, actuating the large lens and camera assembly accurately to track a small moving object such an iris at long range (>10 meters) is challenging due to several factors including that long length of the lens assembly, the weight of the lens assembly, the presence of noise in target (face/iris) localization, the presence of noise in actuator feedback sensors, atmospheric effects, uncertainty in the distance to target, target movement, sensor movement, loss of information due to projection from our 3D world onto the 2D image, variable scene illumination, complex motion of both the tracked object and the camera, full and partial occlusions and changes in appearance of the target. In this report, we limit our focus to mechanics and control methodology and ignore actual image acquisition and issues such as face and iris detection.

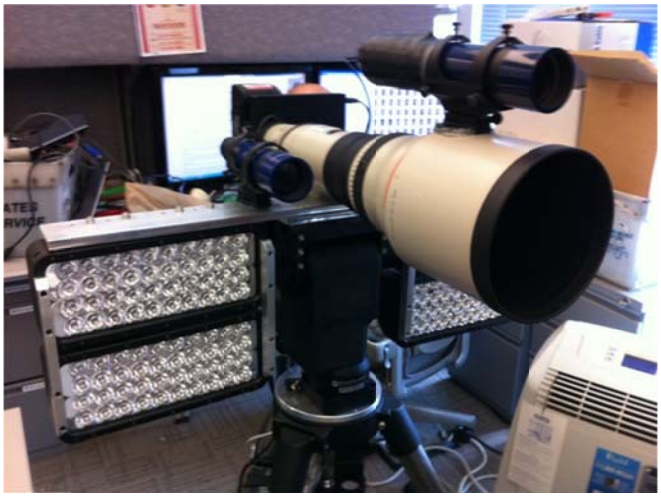

Fig 1. Actual Iris Acquisition System

Pan-Tilt-Zoom Control Model

A block diagram of our proposed system is presented in Fig. 2. Our control model assumes as input a detection location $\{x, y\}$ in the image plane that is acquired from a wide angle view of a scene (either from the PTZ camera at its minimal zoom factor, or another separate camera). This detection could from a haar classifier cascade trained to detect faces, a motion detector, or any other method beyond the scope of this report. These 2D image coordinates are then either fed into a motion model directly that attempts to approximate our 3D world, or 3D world coordinates are first recovered and then used to update our motion model. Using our motion model, we determine predicted locations $\{\hat{x}, \hat{y}, \hat{z}\}$ for where the tracked object should be in the future, and map to our PTZ camera's configuration space $\{\theta, \varphi, f\}$. This predicted configuration is used by the motor controller to steer our high resolution camera to the location of the iris and keep it in view.

Additionally, our system is subject to several real world constraints, such as maximum motor torque, minimum angular resolution, maximum angular velocity, maximum/minimum focal length, and focal length step size.

We limit our consideration and simulation to kinematic constraints only.

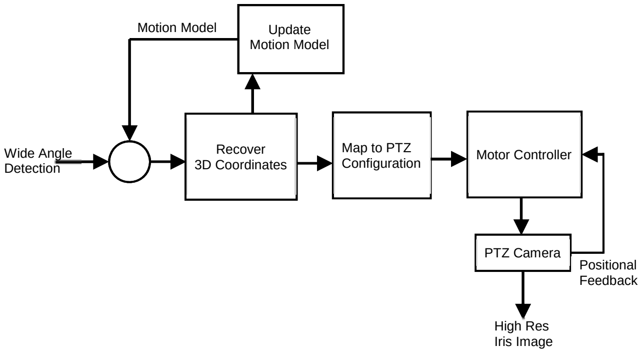

Fig 2. Block Diagram of PTZ Control System

II. Methods

Camera configuration and correlation

There has been extensive research over the years in automated surveillance with cameras in a variety of configurations. Some configurations make use of a single PTZ camera that performs both wide area detection and high resolution acquisition, while others make use of networks of dynamic and static cameras. A challenge when using multiple cameras is to map location coordinates from one camera to another. Steering a camera to a point seen by another camera can be achieved if the 3D position of the object is computable and the relation of the slave camera's position to the coordinate frame is known [5].

The master-slave camera architecture in [4] uses a static, wide field of view master camera to monitor a large area at once, and directs a highly zoomed dynamic camera to approximate target coordinates. Once the target is in the FOV of the slave camera, tracking is performed using the high zoomed images directly. The method used to correlated master camera image coordinates to slave camera Pan-Tilt angles is as follows. First master pixel locations $M_i\{x_i, y_i\}$ are sampled, then the slave camera is manually slewed to center the slave image on the same location, and the corresponding pan-tilt angles $S_i\{P_i, T_i\}$ are recorded. Then simple linear interpolation is applied using the closest two sample points (M_1, M_2).

$$S_j = S_1 + (S_2 - S_1) * \frac{M_j - M_1}{M_2 - M_1}$$

This method has the advantage of being very simple, but it fails to account for the nonlinear process of perspective transformation. Additionally, this method is scene specific, and not useful for a mobile platform.

In [5], an automatic camera-to-camera calibration is performed using the assumption that the ground is approximately planar. A homography, H, is learned as that best matches a set of correspondences in the ground planes of both the master and slave images using RANSAC. To steer the PTZ camera to a point, the system uses a Lucas Kanade tracker to track corners as the camera moves in a outward spiraling star-shaped pattern of pans and tilts. RANSAC is again used to fit an affine transform that best fits the points motion,$(x' - x_0, y' - xy_0)$ for each pan-tilt pair.

$$\begin{pmatrix} p \\ t \end{pmatrix} = T \begin{pmatrix} x' - x'_0 \\ y' - y'_0 \\ 1 \end{pmatrix}$$

Although this allows one to map master image points on the ground plane to pan-tilt angles necessary to steer the slave camera to the desired location, an addition transform is required to map h in the master image to h' in the slave image if varying heights are to be allowed.

$$h' = h\ A \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

This transform $A = (a_0, a_1, a_2, a_2)$ must be learned where (x, y) is the object position in the master image.

Other approaches make use of stereo cameras to recover 3D coordinates, and the approach which we detail below where depth information is recovered by means of an extended Kalman filter that attempts to jointly estimate camera and world parameters.

Motion model and recovery of 3D information using EKF

The Kalman filter is the well-known recursive solution to stochastic estimation from noisy sensor measurements and has been used extensively in tracking. The Kalman filter estimates the process state at some time and obtains noisy feedback estimates. There are two steps to Kalman filters, “time update” and “measurement update” equations. The “time update” step projects the current state and error covariance estimates forward in time. The “measurement update” adjusts the projected estimate by the actual measurement at that time.

Kalman filtering attempts to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

With a measurement $z \in \Re^m$ governed by:

$$z_k = Hx_k + v_k$$

The random variables w_k and v_k are the unknown process and measurement noise vectors which are assumed to by independently distributed, white, Gaussian noise. The (*n* x *n*) matrix *A* is state transition matrix, the (*n* x *l*) matrix *B* is the optional control matrix (which relates the optional control input vector $u \in \Re^l$ to the state), and the (*m* x *n*) matrix *H* relates the state to the measurement vector. The Kalman filter seeks to estimate the *a posteriori* state $\hat{x}_k$ as a linear combination of the *a priori* estimate $\hat{x}_k^-$ and a weighted difference between a measurement z_k and a measurement prediction:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)$$

The (m x n) weight matrix K is known as the Kalman gain and is selected as to minimize the *a posteriori* error covariance of the state. The figure below gives the recursive equations to perform Kalman filtering [7].

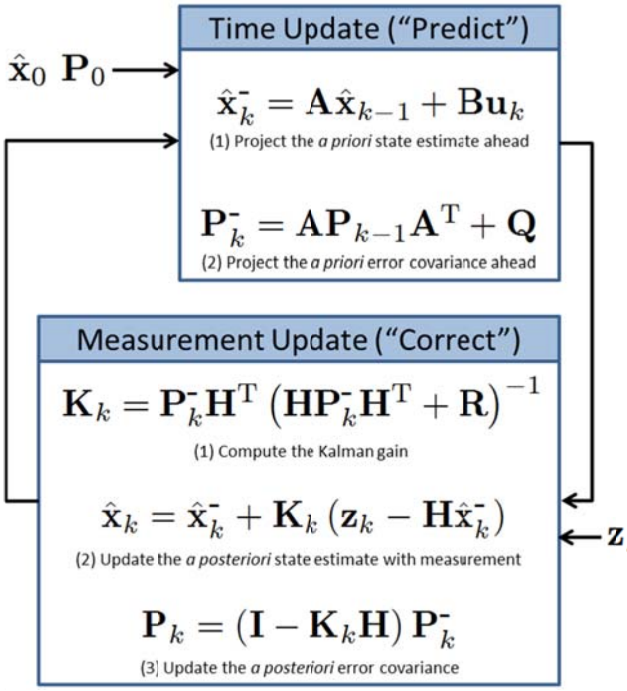
Time Update ("Predict")
$\hat{\mathbf{x}}_0\ \mathbf{P}_0$
$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k$
(1) Project the *a priori* state estimate ahead
$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^{\mathrm{T}} + \mathbf{Q}$
(2) Project the *a priori* error covariance ahead
Measurement Update ("Correct")
$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right)^{-1}$
(1) Compute the Kalman gain
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right)$
(2) Update the *a posteriori* state estimate with measurement
$\mathbf{z}_k$
$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_k^-$
(3) Update the *a posteriori* error covariance

Fig 3. Kalman Filter Update Steps

Kalman filtering has been used in tracking systems to model the 2D position, velocity, and optionally acceleration directly from an object's pixel coordinates (such as those which can be provided from a face detector). This is a reasonable method of tracking which can be used to slew a PTZ camera; however, because this tracker has no concept of depth it fails to accurately predict motions which involve a change in the Z-coordinate.

In [6] an extended Kalman filter to jointly track the object 3D position in the real world, its velocity and the camera focal length, in addition to the rate of change of these parameters. An extended Kalman filter allows for non-linear process $x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$ and measurement $z_k = h(x_k, v_k)$ equations by linearizing about the current mean and covariance estimates. The figure below gives the recursive equations to perform Extended Kalman Filtering [7], where A_k is the Jacobian matrix of partial derivatives of the nonlinear process equation f with respect to the state x_k, W_k is the Jacobian matrix of partial derivatives of f with respect to the process noise w_k, H_k is the Jacobian matrix of partial derivatives of the nonlinear measurement equation h with respect to the state x_k, and V_k is the Jacobian matrix of partial derivatives of the nonlinear measurement equation h with respect to the measurement noise v_k.

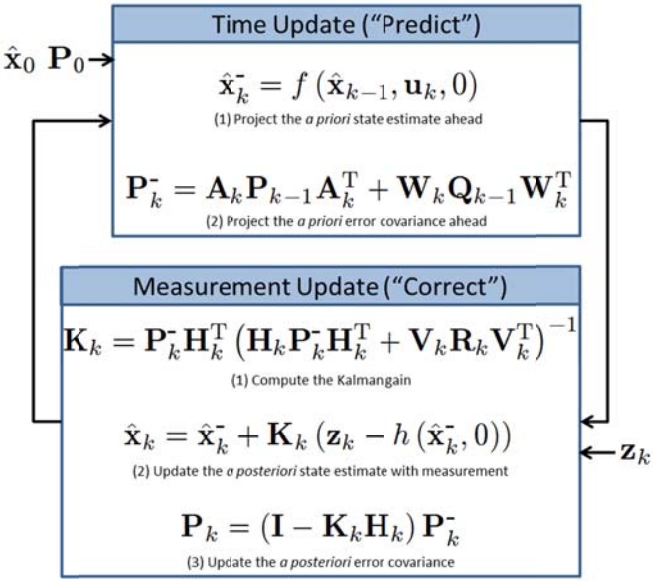
Time Update ("Predict")
$\hat{\mathbf{x}}_0\ \mathbf{P}_0$
$\hat{\mathbf{x}}_k^- = f\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0\right)$
(1) Project the *a priori* state estimate ahead
$\mathbf{P}_k^- = \mathbf{A}_k\mathbf{P}_{k-1}\mathbf{A}_k^{\mathrm{T}} + \mathbf{W}_k\mathbf{Q}_{k-1}\mathbf{W}_k^{\mathrm{T}}$
(2) Project the *a priori* error covariance ahead
Measurement Update ("Correct")
$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{V}_k\mathbf{R}_k\mathbf{V}_k^{\mathrm{T}}\right)^{-1}$
(1) Compute the Kalmangain
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - h\left(\hat{\mathbf{x}}_k^-, 0\right)\right)$
(2) Update the *a posteriori* state estimate with measurement
$\mathbf{z}_k$
$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-$
(3) Update the *a posteriori* error covariance

Fig 4. Extended Kalman Filter Update Steps

With regard to our specific task of recovering 3D position, the process equation remains linear, but the measurement equation, which projects from 3D world coordinates to the 2D image plane is nonlinear. The tracked object is assumed to be a planar patch of known width w and height h located in the world coordinate system at $(X, Y, Z)^T$. This assumption of known size for our planar patch (a valid approximation for a frontal face of an adult of normal dimensions) overcomes the loss of depth information inherent to prospective projection. The camera is modeled as a simple pinhole camera located at the origin of the world coordinate frame, and the camera matrix K is parameterized by a single focal length parameter f. This parameterized camera matrix could be made more complex to account for additional camera parameters.

$$K(f) = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Changing the configuration of $\boldsymbol{q} = (\varphi, \theta, f)^T$ of the camera (where φ is the tilt angle in radians from the vertical, θ is the pan angle in radians from the center of the camera, and f is the current focal length of the camera in meters based on the current zoom setting) is modeled as pure rotations of the coordinate system.

$$R(\varphi, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} cos\,\theta & 0 & -sin\,\theta \\ 0 & 1 & 0 \\ sin\,\theta & 0 & \cos\theta \end{bmatrix}$$

The projection (measurement equation in the EKF) of an object at position $\vec{\boldsymbol{O}} = (X, Y, Z)^T$ is given by:

$$\mathrm{f}(\varphi, \theta, f, \vec{\boldsymbol{O}}) = \begin{bmatrix} \frac{X'}{Z'} \\ \frac{Y'}{Z'} \end{bmatrix}$$

where *X'*, *Y'*, *Z'* are given by the parameterized transformation:

$$\begin{bmatrix}X'\\Y'\\Z'\end{bmatrix} = K(F)R(\varphi,\theta)\overrightarrow{\boldsymbol{O}}$$

Our state vector at time *k* is for the joint camera/object model is a combination of the 3D coordinates of the tracked object $\boldsymbol{O}_k$, the camera parameters $\boldsymbol{C}_k$, and the corresponding velocities $\dot{\boldsymbol{O}}_k$ and $\dot{\boldsymbol{C}}_k$ respectively:

$$\boldsymbol{x}_k = [\boldsymbol{O}_k|\boldsymbol{C}_k|\dot{\boldsymbol{O}}_k|\dot{\boldsymbol{C}}_k]$$

Our linear state transition matrix *A* for constant velocity motion is defined as:

$$\boldsymbol{A} = \begin{bmatrix}\boldsymbol{I}_6 & \boldsymbol{I}_6\\ \boldsymbol{0}_6 & \boldsymbol{I}_6\end{bmatrix}$$

At each time *k*, the observation vector is constructed as follows:

$$\boldsymbol{z}_k = [x_k^p, y_k^p, w_k^p, h_k^p, \varphi_k, \theta_k, f_k]$$

where (x_k^p, y_k^p) are the pixel coordinates of the object in the image plane measured in pixels provided by the face detector, (w_k^p, h_k^p) are the width and height of the detected face in pixels provided by the detector, and $(\varphi_k, \theta_k, f_k)$ are the noisy estimates of the current tilt angle, pan angle, and focal length provided by the camera. The nonlinear measurement equation encapsulating perspective projection is given by:

$$\boldsymbol{x}_k = \left[\mathrm{f}(\varphi_k, \theta_k, f_k, \boldsymbol{O}_k)\middle|\mathrm{f}(0,0,f_k, [W,H,Z_k']^T)\right]^T$$

Configuration Space Mapping

Using the EKF gives an approximate location in the 3D world space of the tracked face. These coordinates must now be mapped into the configuration space in order to get an error function for control of the PTZ. Fortunately, we just need to convert to spherical coordinates in order to get pan and tilt angles.

$$e_\theta = \cos^{-1}\left(\frac{Z_k}{\sqrt{X_k^2 + Y_k^2 + Z_k^2}}\right) - \theta_k$$

$$e_\varphi = \tan^{-1}\left(\frac{Y_k}{X_k}\right) - \varphi_k$$

To calculate the desired zoom level, a desired area in pixels is defined D_a and we use the error function

$$e_f = D_a - w_{proj} * h_{proj}$$

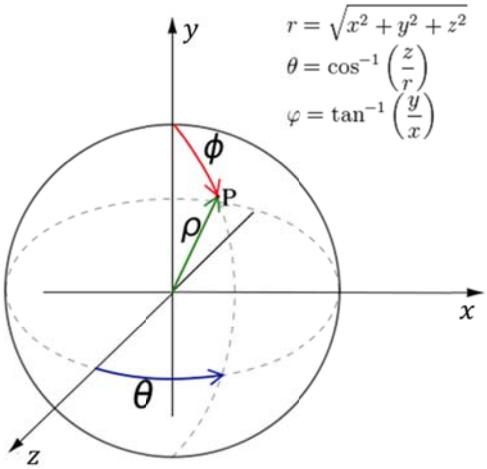
$r = \sqrt{x^2 + y^2 + z^2}$
$\theta = \cos^{-1}\left(\frac{z}{r}\right)$
$\varphi = \tan^{-1}\left(\frac{y}{x}\right)$
y
x
z
ϕ
P
ρ
θ

Fig 5. Spherical Coordinates

Motor Controller

Using the errors $(e_\theta, e_\varphi, e_f)$ as feedback, we seek to control the PTZ camera to minimize these errors as well as minimize settling time. Although our targeted PTZ system has an integrated controller that handles lower level control tasks required to supply sufficient voltage to turn the motors to specified positions, at specified speeds, with necessary torque to overcome attached loads, we still need to specify desired motor velocities. Naïvely, we could use a constant speed controller and simply move each motor in the proper direction so as to reduce our error; however, this would result in terrible performance (tradeoff between terrible settling time and large steady-state error). This method of control is unacceptable, especially for a moving target.

We make use of a proportional-integral-derivative (PID) controller, which is the most widely used controller and is the best controller in the absence of additional information about the underlying process [8][10]. The PID control signal is governed by the equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

A block diagram for a PID controller is shown in the figure below.

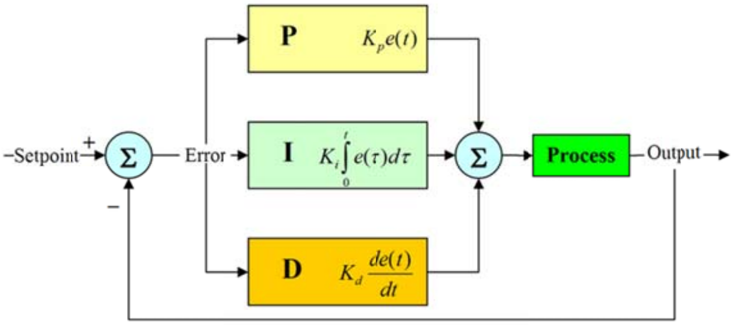

Fig 6. PID Controller Block Diagram [9]

The three terms of the PID controller can be intuitively viewed with respect to time. The proportional (P) term is just the error signal multiplied by a constant gain; it depends only on the current error. High gain can result in instability, while too small of a gain results in unresponsive controller. The integral (I) term depends on accumulated past errors and reduces residual steady-state error. The derivate (D) term is a prediction of future errors, helps reduce overshoot and settling time, but is sensitive to noise. We have one PID controller for each controllable input (PTZ). The gains of our PID controllers were determined experimentally by a manual tuning process outlined in [10]. Additionally, our PID controllers were implemented in software.

III. PTZ Simulation

Using the technique outlined above for control of a PTZ camera, we implemented a 3D simulation in OpenGL of a iris tracker with some simplifying assumptions. We only consider the case of collocated wide-angle (guidance) and telescopic (PTZ) cameras. We limit our telescopic camera to have a fixed zoom setting and thus a fixed focal length. We render a 3D face model (generated previously from a single frontal image using ethnicity and gender specific model). This movement of this face is modeled with 3 independent simple two-state Markov chains (one for each dimension $[X,Y,Z]$). At each time step t_k, with transition probability p_{trans}, transition to the alternate state, where a new random velocity (bounded to prevent superhumanly fast movements) is selected. Velocities are integrated to get current positions, which are limited to keep the disembodied head within a cone of 60 degrees from the origin. The OpenCV face and eye detectors (Viola-Jones haar-cascade classifiers) [11] are used to determine 2D pixel coordinates.

Our model attempts to accurately model the real limitations of actual PTZ hardware. We have a limited angular resolution of 0.002 degrees/second, with a maximum angular velocity of 22 degrees/second. We normalize for scale so that the image size of the face captured is physically anatomically accurate for a human when viewed from a distance of 2-7 meters (ignoring atmospheric effects), with our specified focal lengths.

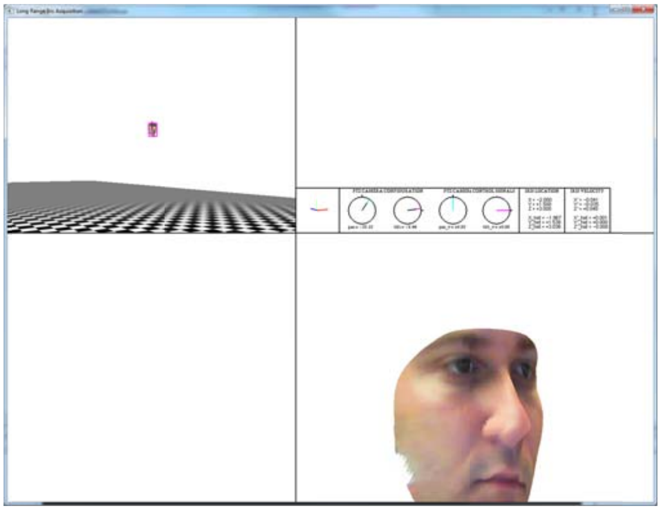

Fig 7. Screenshot from OpenGL PTZ camera simulator showing the wide angle view (top-left) and telescopic view (bottom-right)

IV. Conclusions

We successfully control a PTZ camera in an OpenGL simulation of iris tracking under simplifying assumptions. Future work will involve work on integrating into an operational PTZ system, automatic calibration of camera matrices, and making use of a more intelligent tuning method for our PID controllers. Additionally, we seek to relax the assumption of collocated master and slave cameras—allowing for a fixed master camera with an arbitrary offset from the slave PTZ camera adds flexibility to the configuration. Finally, the iris tracking system outlined above is highly dependent on fast detection (for both face and iris). Utilizing OpenCV's Viola-Jones detector for both is not realistic for real-time tracking with multi-megapixel images.

References

[1] A. Yilmaz, O. Javed, and M. Shah, Object tracking: A survey, ACM Computing Surveys, vol. 38, no. 4, pp. 13+, December 2006. [Online]. Available: http://dx.doi.org/10.1145/1177352.1177355

[2] F.H. Adler, Physiology of the Eye, Mosby, St. Louis, 1965.

[3] J. Daugman, "Probing the uniqueness and randomness of IrisCodes: Results from 200 billion iris pair comparisons," Proc. IEEE, vol. 94, no. 11, pp. 19271935, Nov. 2006.

[4] X. Zhou, R. Collins, T. Kanade, and P. Metes, "A Master-Slave System to Acquire Biometric Imagery of Humans at a Distance," Proc. ACM SIGGM Int'l Workshop Video Surveillance, pp. 113-120, Nov. 2003.

[5] A. Senior, A. Hampapur, and M. Lu. Acquiring multiscale images by pan-tilt-zoom control and automatic multicamera calibration. In IEEE Workshop on Application of Computer Vision (WACV/MOTION'05), volume 1, pages 433–438, 2005.

[6] Murad Al Haj, Andrew D. Bagdanov, Jordi Gonzàlez, F. Xavier Roca: Reactive Object Tracking with a Single PTZ Camera. ICPR 2010: 1690-1693

[7] G. Welch and G. Bishop, "An introduction to the Kalman filter," Tech. Rep. TR 95-041, Univ. of North Carolina, Chapel Hill, 1995.

[8] Nise, S. Norman S, *Control Systems Engineering*, Fourth Edition. 2004.

[9] *PID controller*, Wikipedia. http://en.wikipedia.org/wiki/PID_controller.

[10] Wescott, Tim. *PID without a PhD*. FLIR systems. http://igor.chudov.com/manuals/Servo-Tuning/PID-without-a-PhD.pdf

[11] Intel. Open source computer vision library [Online]. Available: http://www.intel.com/technology/computing/opencv/index.htm