Learning Visual Anticipation: A Top–Down Approach

Vempati Anurag Sai
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
{vanurag}@iitk.ac.in
Advisor: Dr. Amitabh Mukerjee

***Abstract*—This work aims at developing an anticipatory gaze model, in a top-down approach. In the past there have been many models that can track relevant objects from a video making use of computer vision. But one crucial difference is that, a human eye can learn to anticipate the trajectory of an object very quickly in an unsupervised fashion. A sportsperson knows the importance of trajectory estimation better than anyone else. For example, in cricket it is very crucial that the batsman estimates length, bounce and type (swing, spin etc.) of the delivery in a fraction of seconds. In a study by Land & McLeod, 2000 three skilled batsmen facing medium paced deliveries looked at the ball for the first 100-150ms of flight and then made a rapid glance (or saccade) at approximately 50-80% of the ball flight duration to the predicted ball bounce location. This kind of anticipation comes with prolonged training. We aim to build a computational model that could imitate the way humans learn to anticipate the trajectory of fast moving objects.**

***Index Terms*—Anticipatory Gaze; Predictive Vision; Kalman Filter; Linear Regression.**

I. Introduction

In high–speed ball games such as cricket, tennis etc., it is highly important that the player gets a prior estimate as to where the ball is going to land, at what height is he going to make a contact etc. A little difference in the saccade they make during the course of delivery is all there is between aan amateur player and an expert player [1]. This kind of predictive gaze can help the player to get into a stable posotion and also fetch him enough time to search his "shots repository" and come up with the best shot.

Humans track objects of interest around them. While walking around they anticipate future location of people around them so that they wouldn't run into others. Keeping track of other vehicles on road is particularly crucial. The autonomous driving vehicle "Stanley" developed by Stanford University employs a Kalman Filter to keep track of every other vehicle in it's viscinity [5]. Peršе *et al.* have succesfully modelled human motion using Kalman Filter and collision avoidance algorithm thus helping them better the performance of human tracker in highly congested areas. Scuitti *et al.* found out that human gaze is tightly connected to the motor resonance system. While a sequential task is being performed, the observer was found to anticipate the task's goal rather than just track the demonstrator's motion.

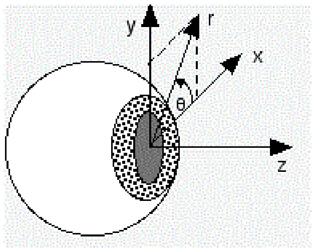

Fig. 1: The coordinate system. [Courtesy:IUSO vision sciences group]

All this research indicates as to how important it is for humans to anticipate intention or position of salient objects in their day–to–day life. In such scenarios, the human gaze is mostly saccadic, continuously predicting where they have to look next. This work aims to develop a computational model that can visually anticipate like humans do. We will be restricting ourselves to a simplistic scenario of anticipating a ball's future location when it's moving at very high speeds.

II. Model Description

Since, the motto is to learn to predict a ball's trajectory in unsupervised fashion, we will be first building a dataset for the training phase.

A. Dataset

Note: We will be assuming a right-handed coordinate system where, z–axis points in the viewing direction and x–axis is parallel to line joining the two eyes as shown in Fig.1

The dataset consists of a ball bouncing off the walls and floor as viewed from different viewpoints. The ball's speed and release direction are randomly chosen with a swing/spin components incorporated in few of the videos. Each video contains roughly about 150 frames.

We first model the physical world. Let's define a 'state' as a tuple consisting of position, velocity and accleleration in three orthogonal directions.

$$S_t^G = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \\ v_x(t) \\ v_y(t) \\ v_z(t) \\ a_x(t) \\ a_y(t) \\ a_z(t) \end{pmatrix}$$

where, $[x(t), y(t), z(t)]$, $[v_x(t), v_y(t), v_z(t)]$, $[a_x(t), a_y(t), a_z(t)]$ are the ball's posotion, velocity and acceleration respectively w.r.t ground coordinate system, at time t.

Each state evolves from it's previous state by applying a kinematic transformation F as follows.

$$S_{t+1}^G = F * S_t^G \tag{1}$$

where,

$$F = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{(\Delta t)^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{(\Delta t)^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{(\Delta t)^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The swing component can be introduced by adding a velocity component proportional to velocity in it's orthogonal direction. For example, making entry at 4th row, 5th column non–zero, we can add velocity component in x–direction, proportional to velocity in y–direction. Though this is a crude way to acheive swing, it will serve our job.

We will have a different Transition matrix F_{bounce} when the ball bounces off the floor or wall. For example, if the ball is bouncing off the floor, the velocity in y–direction is reversed and then F is applied. So, $F_{bounce} = F * B$ where, B is an Identity matrix of size 9×9 with entry at 5th row, 5th column replaced by -1.

Basically, we can boil down any physical phenomenon in a crude way, to a variation in the transition matrix F. This rough approximation is acceptable since, Kalman Filter takes care of such issues by introducing process noise into it's design which we will talk in detail later.

On top of the physics model, "viewpoint transformation" is applied. This can be acheived as follows:

$$S_t^{new} = R_G^{new} * S_t^G \tag{2}$$

where, R_G^{new} is the rotationmatrix from ground reference frame to the new reference frame. We don't consider the translation vector from ground to new reference frame because, in our kinematics equation, the states and Transition matrix F remain invariant to translation. Only the x,y and z in state are shifter by some constants. Since, we don't explicitly use state entries anywhere in our algorithm and F remains unchanged, we need not use translation vector.

We will be applying one last transformation based on the geometry of Optics. i.e., "farther objects are smaller". This can be acheived as follows:

$$x^{visual}(t) = f * x^{new}(t)/z^{new}(t) \tag{3}$$

$$y^{visual}(t) = f * y^{new}(t)/z^{new}(t) \tag{4}$$

$$r^{visual}(t) = f * R/z^{new}(t) \tag{5}$$

where, $x^{visual}(t), y^{visual}(t)$ and $r^{visual}(t)$ are the position and size of the ball (of physical radius R) as seen from the new viewpoint with an eye/camera of focal length f.

B. Parameter Extraction

For each video in the dataset, the ball's position and size has to be recovered in every single frame. The data thus extracted will be used for training the model. Though there are several methods based on segmentation and optical flow that can acheive this, since we are specifically looking for a ball, Hough Transform will perform extremely well.

A Canny Edge detector is ran to extract out all the edges. This gives us rough outline of every object. A Hough transform first creates a 3–D matrix of size (num_rows, num_cols, max_radius) where, 'num_rows', 'num_cols' are number of rows and columns in each frame. 'max_radius' is the maximum size of the ball that Hough Transform can detect.

Now, for each point detected by the Canny edge detector, all possible circles that can pass through that point are up–voted. Say, if a circle of radius 5 with center at (210, 154) can pass through an edge point, the entry in the 3–D matrix corresponding to these values is raised by one. As it can be seen, once this process is finished, the entry that corresponds to the ball's position and radius gets maximum number of votes. Thus, we extract the necessary parameters. The position of the ball and it's size in each frame is entered (as a triplet) into a file for further use.

C. Learning the Transition Matrix

For the system to be able to predict the future states, it should come up with a good estimate of the Transition matrix F. We will be using the data collected from the training videos.

Since there are "viewpoint transformation" and "Optical transformation" on top of the "kinematic transformation", we need to revert both of them. The set of transformations on each triplet (x, y, r) looks as follows:

$$z^{new}(t) = f * R/r$$

$$x^{new}(t) = x * z^{new}(t)/f$$

$$y^{new}(t) = y * z^{new}(t)/f$$

At this point we just have position vector. But we need velocity and acceleration to form the state vetor. So velocity is estimated as $v = \Delta x/\Delta t$ and acceleration by $a = \Delta v/\Delta t$ where, Δt is the duration of each frame in the video. This kind of approximate estimates will only be used during the learning phase since, the regression problem takes care of the errors. We won't be using such estimates while predicting future position of the ball since that might result in cascading effect due to error propogation from one state to the next. More on this later.

This will be followed by inverting the viewpoint transformation by multiplying by $(R_G^{new})^{-1}$ which is same as $(R_G^{new})^T$ (Transpose of the Rotation Matrix).

$$S_t^G = (R_G^{new})^T * S_t^{new}$$

Here, R_G^{new} is found by evaluating the transformation that can transform the floor in visual domain to a plane horizontal to x–z plane in Ground reference frame. Research indicates that humans on the other hand don't rely on visual inputs alone to determine their head orientation (which decides the viewpoint). Healthy individuals were found to be able to position themselves parallel to gravity within 0.5^0 even in the dark [6].

From here on, learning the state transition matrix can be boiled down to a set of simple linear regression problems. Each learns a row of the matrix. The i^{th} problem looks as follows:

$$S_{next}^G(i) = S_{current}^G * (F_i)^T \tag{6}$$

where, $S_{next}^G(i)$ is the vector of size $N-1$ defined as, $[S_2^G(i), S_3^G(i), ..., S_N^G(i)]$ and $S_{current}^G$ is a matrix of size $(N-1) \times 9$ defined as,

$$S_{current}^G = \begin{pmatrix} \text{———} (S_1^G)^T \text{———} \\ \text{———} (S_2^G)^T \text{———} \\ \vdots \\ \text{———} (S_{N-1}^G)^T \text{———} \end{pmatrix}$$

N is the number of training examples in the dataset and $S_t^G(i)$ is the i^{th} element of the vector S_t^G. F_i is the i^{th} row of the Transition matrix F.

This regression problem has a Least-Squares estimate given by:

$$(F_i)^T = \left(\left(S_{current}^G\right)^T S_{current}^G \right)^{-1} \left(S_{current}^G\right)^T S_{next}^G(i) \tag{7}$$

Similarly, the rest of the rows of the Transition matrix F are estimated.

D. Kalman Filter

Now that we have the Transition Matrix F, we could have estimated the future position of the ball by repeatedly multiplying the present state with F. But, there are several problems with this approach:

- Since there are errors associated with the ball's parameter detection (position and size), these get transferred to the velocity and acceleration estimates we might use to build the state. And, when we multiply the state several times by the Transition matrix F, the error gets compounded and results in a bad prediction.
- As the Hough Transform searches the parameter space in discrete steps, this results in a slight error in position and size of the ball. This is not a big problem in the case of position but a slight change in size results in a lot of difference in the z–estimate. So, the estimates of velocity and acceleration in z–direction are even worse.

On the other hand, there are several benefits in using a Kalman Filter, such as:

- Only the position estimates are taken as the input. The velocity and acceleration are taken care of.
- It considers measurement noise. So slight errors in position estimates are easily removed.
- We need not have an accurate Transition Matrix because it takes process noise into consideration.
- Several cycles of prediction can be run before next measurement update.
- The design is robust and accurate.

Kalman Filter basically takes noisy measurements and comes up with statistically optimal estimate as to what the actual state might be. With every new measurement the prediction gets better. More formally, Wikipaedia defines Kalman filter as follows: The true state at time t is evolved from the state at $(t-1)$ according to

$$S_t = F_t S_{t-1} + B_t u_t + w_t \tag{8}$$

where, F_t is the state transition model which is applied to the previous state S_{t-1}, B_t is the control-input model which is applied to the control vector u_t and w_t is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_t.

At time t an observation (or measurement) z_t of the true state S_t is made according to

$$z_t = H_t S_t + v_t \tag{9}$$

where, H_t is the observation model which maps the true state space into the observed space and v_t is the observation noise which is assumed to be zero mean Gaussian noise with covariance R_t.

In our model, F_t is the state transition matrix F, $B_t = 0$ and the rest are chosen as follows:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

R_t = $0.5 \times I_{3\times3}$ and process noise Q_t is initialized as $0.05 \times I_{9\times9}$ where, I is an Identity Matrix. P is the state covariance matrix initialized as $1000 \times I_{9\times9}$.

The algorithm [4] is a recursive estimator. Each iteration has a measurement update routine and prediction routine.

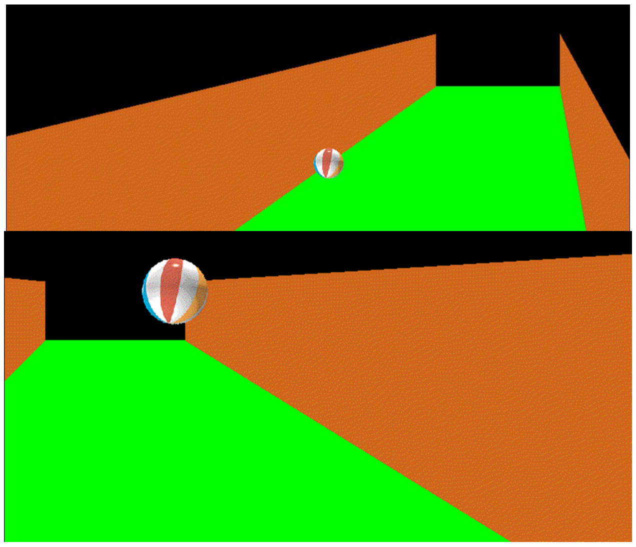
Fig. 2: Sample frames from two different training videos

Measurement Update Routine:

$$Y_t = z_t - H_t * S_t \quad (10)$$
$$T_t = H_t P (H_t)^T + R_t \quad (11)$$
$$K_t = P_t (H_t)^T T_t^{-1} \quad (12)$$
$$S_t = S_t + K_t Y_t \quad (13)$$
$$P = (I_{9\times 9} - K_t H_t) P \quad (14)$$

Prediction Routine:

$$S_{t+1} = F_t S_t + B_t u_t + w_t \quad (15)$$
$$P = F_t P F_t^T \quad (16)$$

where, S_{t+1} is the predicted state at time $t+1$. The prediction routine can be run several times in each iteration to predict further.

III. Results

A. Dataset

Fig.2 shows two sample frames from different training videos with randomly set viewpoints, velocities and spin/swing components. In total, 4 videos each consisting roughly around 150 frames are used in training phase and one video for test phase. The videos are of resolution 860 $\times$ 360.

B. Parameter extraction

Fig.3 shows the output of Canny edge detector on a single frame. Fig.4 shows the ball detected by the Hough Transform (circled) in two different frames. It performs quite good but there can be slight errors as in Fig.4(b).

C. Learnt Transition Matrix

The learnt Trasition Matrix F is close to the ideal transition matrix with an rms error of 0.0094.

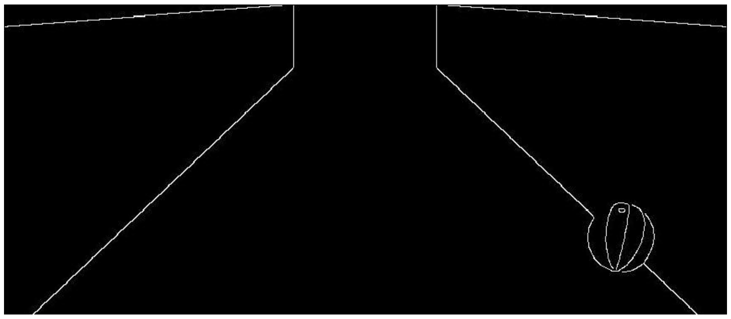
Fig. 3: Canny Edge detector output

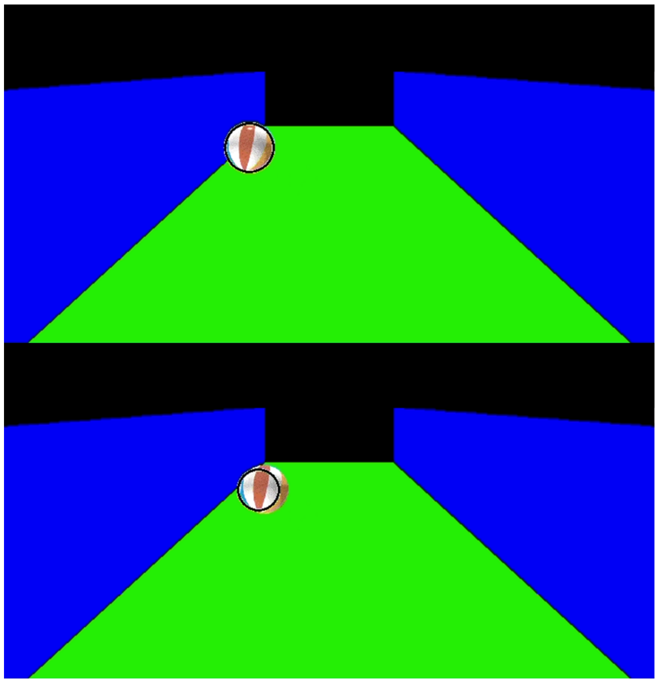
Fig. 4: Ball detected by the Hough Transform (a)successful and (b)errroneous

D. Prediction

The prediction worked pretty good with an rms error of 12.36 pixels in position. Fig.5 shows the predicted trajectory at one particular instant of the test video.
video link: http://www.youtube.com/watch?v=psldAbFzlHk

References

[1] Land, Michael F., and Peter McLeod. “From eye movements to actions: how batsmen hit the ball.” Nature neuroscience 3.12 (2000): 1340-1345.
[2] Sciutti, Alessandra, et al. ”Anticipatory gaze in human-robot interactions.“Gaze in HRI from modeling to communication” workshop at the 7th ACM/IEEE international conference on human-robot interaction, Boston, Massachusetts, USA. 2012.
[3] Perse, Matej, et al. “Physics-based modelling of human motion using kalman filter and collision avoidance algorithm.” International Symposium on Image and Signal Processing and Analysis, ISPA05, Zagreb, Croatia. 2005.
[4] http : //en.wikipedia.org/wiki/Kalman_filter
[5] Thrun, Sebastian, et al. ”Stanley: The robot that won the DARPA Grand Challenge.” Journal of field Robotics 23.9 (2006): 661-692.

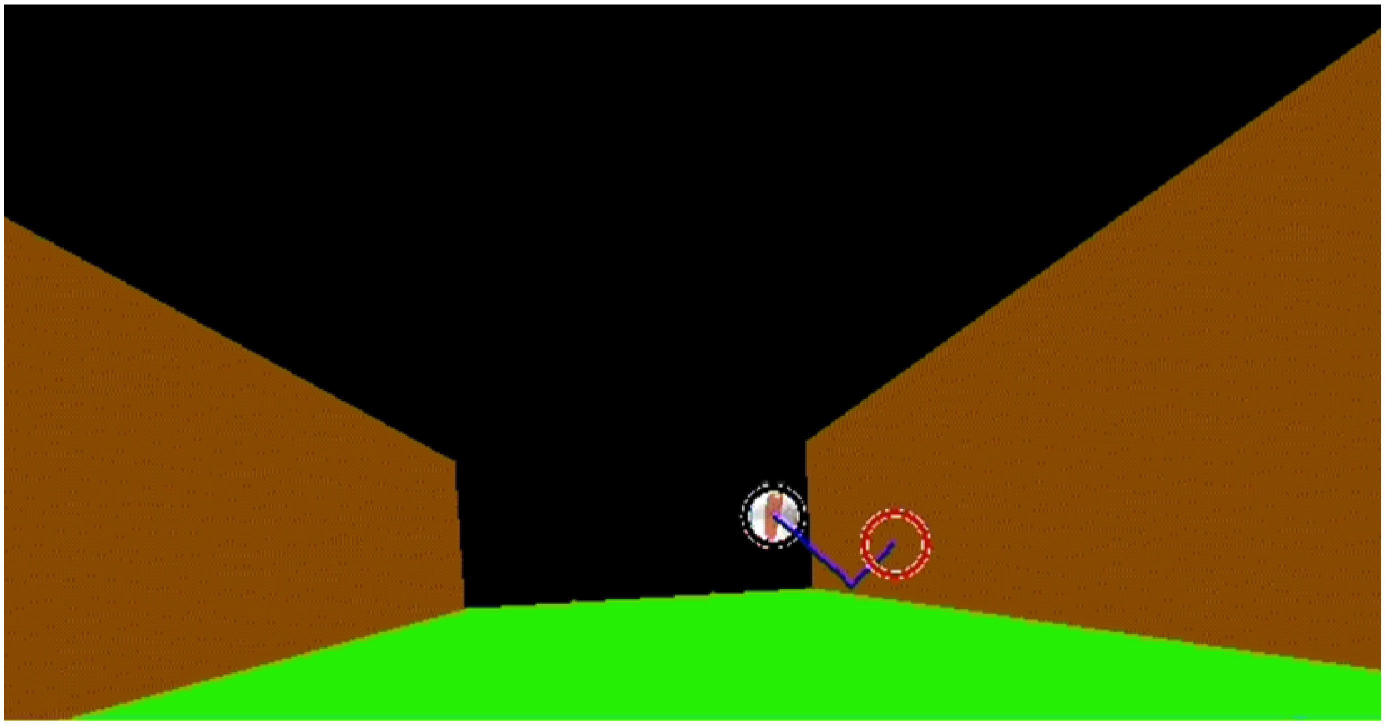
Fig. 5: Ball's estimated trajectory (10 cycles of prediction in 'blue') and size (in 'red' circle)

[6] Horak, Fay B. "Postural orientation and equilibrium: what do we need to know about neural control of balance to prevent falls?." Age and ageing 35.suppl 2 (2006): ii7-ii11.