

Robocanes Goalie Agent: Behavior and Ball Prediction System

Joseph Masterjohn

Abstract

This report is a collection and analysis of the methods and ideas implemented for the final project for the class CSC688: Autonomous Robotic Systems, taught at the University of Miami in the Fall of 2014. The project was focused around creating a rational agent with the role of goal keeper within the Robocanes Robocup Simulation League software. The two main facets of the project were focused on sensing the ball position and velocity, and creating a model for the agent's behavior. Various techniques used for each of these distinct problems are studied and compared.

1 Ball Prediction

The scenario put forth for the challenge was that of a "Goal Kick". That is, a single stationary striker opponent kicking the ball towards a single goalie with the intent of scoring a goal. Only two agents exist on the field, the striker and the goalie. Upon preliminary study of the scenario, one caveat stood out: in order to successfully position itself so to block the oncoming goal kick, the goalie agent needed an accurate prediction the position of the ball on its predetermined trajectory when it crossed the goal line. In order to make this prediction as well as other useful sensory data about the ball's action (ETA of ball arrival, velocity of the ball, etc.), the kinematics of the ball needed to be modeled accurately. The existing Robocanes software platform already implements a fairly accurate representation of the ball's position (BallPos), but does not provide any accurate model of the velocity of the ball. Thus two methods were attempted with the intent of providing this information.

1.1 Regression Model

Upon initial analysis of the accuracy of the existing ball position in the software, it was decided that the first and most elementary approach to predicting the ball velocity would be to compute a simple linear regression on the existing ball positions. This, obviously simplified, method operates under a few assumptions that are not accurate to the realm of the simulator:

- $|v| \neq 0$, for v the velocity of the ball
- $\forall \delta t \in \mathbb{R}, x'([t, t + \delta t]) = c, c \in \mathbb{R}$ (i.e. - constant velocity)

This model does not take into account the complex motion of the ball, but it does serve as a starting point for an efficient way to model the ball velocity. In order to do the initial regression, the Thiel-Sen algorithm was used. This method for calculating the linear trend of a set of points is not as sensitive to outliers as other methods [1]. The full algorithm for determining the velocity follows:

```

function VELOCITYREGRESSION( $S = [(x_1, y_1, t_1), \dots, (x_n, y_n, t_n)]$ )

     $slopes = \emptyset$ 

    for all  $p_i, p_j \in S : p_i \neq p_j$  do

         $m = \frac{(y_i - y_j)}{(x_i - x_j)}$  ▷ slope of two unique points in the set

         $slopes = slopes \cup m$  ▷ Include m in the global set

    end for

     $sort(slopes)$ 
     $M = selectMedian(slopes)$  ▷ M is the median of all of the slopes

     $intercepts = \emptyset$ 

    for all  $p_i \in S$  do
         $b = y_i - Mx_i$ 
         $intercepts = intercepts \cup b$ 
    end for

```

```

    sort(intercepts)
    B = selectMedian(intercepts)

    v = (1, M + B) - (0, B)      ▷ Create the velocity vector v from two
                                ▷ points on the line, x = 1 and x = 0

    vRough = pn - pi            ▷ Rough direction of the velocity

    if v * vRough < 0 then        ▷ If v isn't in the right direction
        v = -v
    end if

    v = v *  $\frac{|vRough|}{|v|}$       ▷ Scale v to the distance between first and last
    v = v / (tn - ti)          ▷ Scale v to be accurate to the timestep

    return v
end function

```

The gist of the algorithm:

1. Let the slope of the regression line, M , be the median of all slopes calculated between unique pairs of points in S
2. Let B be the median of all calculated y-intercepts of all points in S
3. Choose the orientation of the vector defining the line based on the rough direction of the points, given by the vector between first and last points.
4. Scale the vector to accurately represent the true speed of the ball, given the rough distance and accurate timestep

Figure 1, Figure 2, and Figure 3 show a progression of 3 frames with the predicted ball velocity (computed using the regression model) highlighted in blue. The predicted goal line intercept is highlighted in yellow, on the goal line.

1.2 Kalman Filter Model

While the first approach succeeded in giving an estimate of the ball velocity accurate enough to predict the goal line crossing and ETA, a second



Figure 1: Frame 1 (Regression)



Figure 2: Frame 2 (Regression)



Figure 3: Frame 3 (Regression)

approach was implemented in order to compare a well known and studied method (Kalman Filtering) with a completely ad-hoc method (Regression). This decision was made with the additional foresight that the success of the linear regression model might be dependant on the magnitude of error realized in the simulation. Thus a method robust enough to predict the state of the ball for a real world environment would then be integral.

The Kalman Filter is a well known and studied method of estimating un-measurable variables through a series of noisy measurements , observed over time [2]. The formulation of the Kalman Filter matrices were as follows:

$$x = [x, y, x', y']$$

The state vector containing position and velocity

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix}$$

The covariance matrix, initially our measurement of x and y are believed to

be accurate, but we have no confidence in the velocity.

$$F = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The state transition matrix based on the kinematic equations:

$$x_{i+1} = x_i + v_{ix}\delta t$$

$$y_{i+1} = y_i + v_{iy}\delta t$$

$$v_{(i+1),x} = v_{ix}$$

$$v_{(i+1),y} = v_{iy}$$

Here we see an initial simplification that will inevitably affect the accuracy of this estimation. The velocity is again assumed to be constant between small time steps, as in the regression model. If we assume deceleration due to friction is negligible, especially during a hard kick or a kick that spends a considerable amount of its trajectory in the air, this will not create terrible inaccuracy.

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

Our measurement matrix. We obtain no external information about the velocity.

$$R = \begin{bmatrix} 0.95 & 0 \\ 0 & 0.95 \end{bmatrix}$$

Our model of the measurement noise. We assume the measurements are confident with a 5% error factor.

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

Our process noise matrix. We assume an error of 1% in each variable.

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The 4x4 Identity matrix used in the update of the Kalman Matrices.

The update of the matrices at each timestep follows the standard Kalman Filter update process:

- $z = \text{measurement}(x, y, x', y')$
- $y = z - Hx$
- $S = (HP_iH^t) + R$
- $K = P_iH^tS^{-1}$
- $x_{i+1} = x_i + Ky$
- $P_{i+1} = (I - KH)P_i$

The measurement stage.

- $x_{i+1} = Fx_i$
- $P_{i+1} = FPF^t + Q$

The prediction stage.

From this update, the calculation of the velocity is simple:

$$v = x_i([2, 3])$$

$$v = \frac{v}{t_n - t_i}$$

The velocity is taken directly from the Kalman prediction and then scaled to the timestep.

The result of the estimated velocity, predicted by the Kalman filter can be seen in Figure 4, Figure 5, Figure 6, and Figure 7. Again the velocity is highlighted in blue, and the predicted goal crossing is highlighted in yellow.



Figure 4: Frame 1 (Kalman)



Figure 5: Frame 2 (Kalman)



Figure 6: Frame 3 (Kalman)



Figure 7: Frame 4 (Kalman)

1.3 Results

In order to provide a quantifiable analysis of the two methods used, I computed the accurate frame-by-frame velocity using the groundtruth of the simulation. Then I calculated a Root Mean Squared Error metric for the estimated velocity at each frame of a kick. Let v_i be the groundtruth velocity at frame i , and let v_{ik} and v_{ir} be the estimated velocity from the Kalman Filter and Regression method respectively. The calculated errors are as follows:

$$error(magnitude) = \sqrt{\frac{1}{n} \sum_0^n (|v_i| - |v_{ik}|)^2}$$

$$error(angle) = \sqrt{\frac{1}{n} \sum_0^n |acos(v_i * v_{ik})|^2}$$

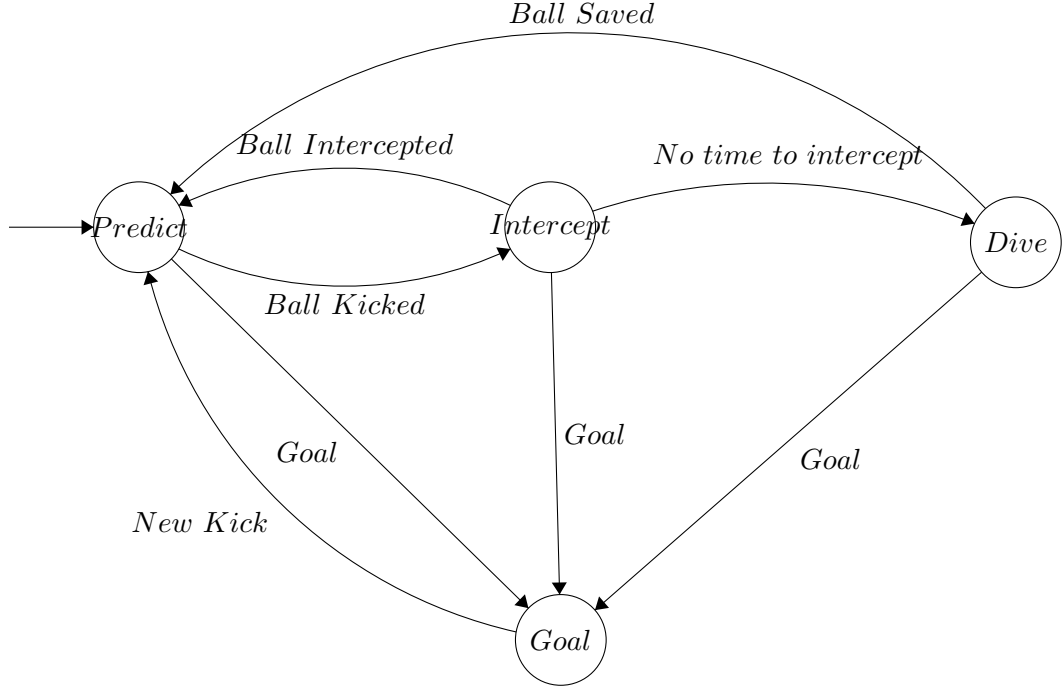
The error results are summarized in the table below:

<i>Method</i>	RMS Mgnitude Error (m/s)	RMS Angle Error (rad)
Regression	0.688517	0.209669
Kalman Filter	0.505363	0.218454

As expected, the Kalman Filter made a slight improvement in the magnitude estimation. Both methods produced an average error of 0.2 radians in the direction of the velocity. At a distance of 5 meters from the goal line, this produces a horizontal uncertainty of around 1 meter in each direction. This uncertainty converges the closer the ball is to the goal line. It is still questionable whether this amount of uncertainty is tolerable for the agent to make a decision. Some initial methods to quell this error were to average the predicted goal crossings across several frames and use that as the estimate of the goal crossing. This slightly improved the accuracy of the prediction.

2 Robot Behavior

The systematic behavior of the agent can be described by a finite state machine:



This simple observation-action model is the whole basis of the goalie agent. There were some caveats to determining when the transitions happen, for instance: When is the ball actually kicked? What is the proper time to dive for the ball? Some of these behavioral questions had to be answered analytically, and others with heuristic methods.

The ball velocity module described in the above Section 1 provides a heuristic model to provide the instantaneous moment of a kick. It does this by doing a rough calculation of the instantaneous acceleration of the ball. This in itself is inaccurate for the scenario because at the beginning of every kick, the ball is teleported to a new location for the beginning of a new kick. Thus at that precise moment the ball, numerically speaking, has an infinite velocity. This and other small hitches had to be dealt with each with conditional logic for their respective corner cases.

In order to determine the correct time to dive for the ball, two pieces of logic were used. The first was a rough estimate of the duration of the fall. If the agent needs to dive for the ball, it needs to have enough time to get to the ground before the ball arrives. This value was experimentally found by running multiple simulations with different diving motions. The second

piece of knowledge used by the agent was the ETA of the ball, provided by the ball velocity module. This time was compared to the time it would take for the agent to reach the estimated goal crossing position. If there was enough time, the agent would continue to walk towards it, if not the agent would transition to the dive state.

The rest of the transitions were based on the game state of the simulation. The goalie predicted an optimal position to stand before the ball was kicked. This positioning proved to be crucial to the agents ball saving abilities. If the goalie was on the opposite side of the goal when a shot was kicked, he would typically not have enough time to maneuver to the accurate predicted position nor have the extension to dive to the position. Thus an accurate predicted standing position is very beneficial to the agent. The position that the agent currently predicts is based on inference from visual inspection. At a certain radius, it is beneficial to stand at the side of the goal opposite to the striker, as the majority of shots from those positions tended to enter the goal with that trajectory. Smaller than a certain threshold the optimal place to stand would be on the same side of the goal as the striker. Each of these deductions come from intuition, and thus are sub-optimal in terms of the capability of the agents.

3 Results and Future Work

At the time of this report, the collaborative agent constructed by myself and Julian Jarret faired the best of all agents in the CSC688 class, saving 25% of the sample goal kicks in competition and 34% of kicks after slight changes in the testing system allowed for our Kalman Filter prediction model to operate in the test environment. The combination of accurate ball velocity prediction with predictive goalie positioning proved to be a sufficient first attempt at improving the existing agent infrastructure for the goalie, improving the scored-to-saved ratio by ~15%.

Some additional ideas for improvements as well as additional work include, but are not limited to:

- Statistical Learning Model for predictive goalie positioning
- Creating new save motions (dives, splits, putting hands out, etc.) by optimizing joint values during kick scenarios

References

- [1] Sen, Pranab Kumar (1968): *"Estimates of the regression coefficient based on Kendall's tau"*, Journal of the American Statistical Association 63: 13791389, JSTOR 2285891
- [2] Ramsey Faragher *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation* IEEE Signal Processing Magazine [128], SEPTEMBER 2012