2002

Improbability Filtering for Rejecting False Positives

Brett Browning
Carnegie Mellon University, brettb@cs.cmu.edu

Michael Bowling
Carnegie Mellon University, mbac@andrew.cmu.edu

Manuela M. Veloso
Carnegie Mellon University, veloso@cs.cmu.edu

Published In
.

Improbability Filtering for Rejecting False Positives

Brett Browning, Michael Bowling, Manuela Veloso
School of Computer Science, Carnegie Mellon University
{brettb, mhb, mmv}@cs.cmu.edu

***Abstract*—In this paper we describe a novel approach, called improbability filtering, to rejecting false-positive observations from degrading the tracking performance of an Extended Kalman-Bucy filter. False-positives, incorrect observations reported with a high confidence, are a form of non-Gaussian white noise and therefore degrade the tracking performance of an Extended Kalman-Bucy Filter. Improbability filtering removes false-positives by rejecting low likelihood observations as determined by the model estimates. It offers a computationally fast and robust method for removing this form of white noise without the need for a more advanced filter.**

We describe an application of the improbability filter approach to Extendend Kalman-Bucy filters for tracking ten robots and a ball moving at speeds approaching 5 $m.s^{-1}$ both accurately and reliably in real-time based on the observations of a single color camera. The environment is highly dynamic and non-linear, as exemplified by the motion of the ball which varies from free rolling under friction, to rolling up 45° inclined walls at the boundary, to being manipulated in unpredictable ways by a mechanical apparatus on each robot. The sensing apparatus, a camera and color blob tracking algorithms, suffers from the usual noise, latency, intermittency, as well as from false-positives caused by the misidentification of an observed object with a non-negligible likelihood.

I. Introduction

The Kalman-Bucy filter is a widely used algorithm for tracking observable parameters in linear systems under the presence of stochastic noise [4, 5]. In short, it has been described as the optimal tracking algorithm [9] for parameters in the presence of Gaussian noise. The Extended Kalman-Bucy Filter (EKBF) extends the algorithm to include non-linear systems whereby the model is linearised about the state estimate [11].

The filter algorithm is based on the fundamental assumption that the parameters of the observations and dynamics have a Gaussian probability distribution. In other words, any noise components in the system are Gaussian in nature. Although noise is commonly not Gaussian in nature, with suitably chosen parameters, the filter is often robust to moderate violations of the Gaussian noise assumption.

This paper describes an extension of the EKBF algorithm to remove white noise, in the form of false-positive observations, from degrading the performance of the filter. False-positives, where incorrect observations are reported with high confidence, violate the Gaussian assumption underlying the EKBF. Modifying the EKBF parameters can compensate for false-positives but at the expense of tracking ability. We describe a new approach, called Improbability Filtering (ImpF), developed to overcome this problem. ImpF uses the state and co-variance estimates of the EKBF to filter out unlikely observations, thereby removing false-positives and enabling the EKBF to track reliably. We describe an application of this algorithm to small size robot soccer where the tracking module must reliably track ten fast moving robots and the ball at the camera frame rate (~60Hz). The highly dynamic nature of the environment, the presence of false-positives caused by misidentifications, and the need for fast computations provide an ideal testing grounds for ImpF.

The following section briefly describes the robot soccer application that is the basis for the tracking system. Section III describes the EKBF's that we implemented to track different objects and section IV describes the improbability filter algorithm, its approach and implementation along with the results of its performance. In section V, we discuss these results and conjecture where the improbability filter could be used in other applications. Section VI concludes the paper.

II. Tracking In Robot Soccer: The Problem

RoboCup robot soccer is an exciting and challenging robot domain aimed at advancing robot intelligence and control [7]. This paper focuses on the tracking module implementation in a small size robot team where two teams of five robots compete autonomously in a game of soccer with an orange golf ball on a carpeted, ping-pong table-sized field with 45° inclined walls along the boundaries [1]. Figure 1 shows the system structure with an illustration of two robots from either team.

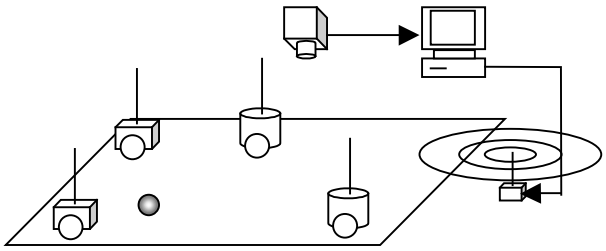

Figure 1. Each robot is observed by the camera and tracked via colored markers on top of its cover.

Each robot receives radio commands from an off-field PC. The PC generates the commands based on its strategic and tactical algorithms, which in turn derive information about the world from an overhead camera. The overhead camera provides interlaced fields at a rate of 60Hz. Each field is processed using the CMVision color recognition system [2] and the results are reported along with an estimated confidence measure. Thus, each object is reported as:

$$object_j^{obs} \Rightarrow \left\langle x_j^{obs}, y_j^{obs}, \theta_j^{obs}, C_j^{obs} \right\rangle$$

Here C^{obs} is the confidence of the reported observation over the range 0.0 to 1.0, j refers to the object being tracked be it one of the ten robots or the ball. For the opponent robots and the ball the orientation θ^{obs} has no meaning.

Vision sensing suffers from a number of the usual problems associate with sensors. Data is noisy, intermittent, and has significant latency. Moreover, robot covers have patterns for individual robot identification which when viewed by the overhead camera are occasionally misidentified with one another causing the reported robot locations to jump around between frames. Misidentifications, or false-positives, act as a white noise source on top of the usual Gaussian noise in the sensing stage. Such white noise causes havoc with many tracking algorithms due to the deviations from the Gaussian noise pattern that they cause.

To accurately control the robots at speed, the tracking system needs to account for the deficiencies in the vision output. Additionally, it needs to derive velocity information that cannot directly observe. Finally, it needs to provide facilities to predict, with appropriate confidence measures, where the objects on the field will be at some given time in the future. Although there are many statistical tracking algorithms that have been developed, the Kalman-Bucy filter has proved to be one of the most capable filters. The KBF and its extended variant for EKBF non-linear systems, are widely used [3], have a solid and well-understood mathematical basis, and are computationally tractable when used in such a system as the one discussed here. As such, the EKBF represents a good choice for a tracking mechanism and was the one used for the robots described in this paper and is discussed next.

III. Tracking with the Kalman-Bucy Filter

Each of the eleven objects, two teams of five robots and the ball, is tracked using an independent EKBF. Multiple EKBF's are used due to the non-linear nature of the tracking problem. Robot kinematics are inherently non-linear as is the motion of the ball both along the inclined walls and along the field surface. Multiple independent EKBF's are used based on the assumption that the observations of different objects are independent. This is a reasonable assumption that helps preserve limited computational resources. There are two filter types; the robot EKBF and the ball EKBF, where the difference is due to the different dynamics of the object being tracked. We briefly describe the EKBF equations that are the basis for the filters, which are an extension on our previous EKBF's [8]. For a more detailed discussion of EKBF's refer to [3, 9, 6, 11]. The symbols used here are identical to [11].

In the following discussions, the state estimate will be represented by $\tilde{x}_k$ and the state covariances by P_k. Two stochastic difference equations model the system dynamics and observations, respectively, as:

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \text{ and } z_k = h(x_k, v_k)$$

Here $\mathbf{z_k}$ is an observation of the system at step k, u_k is the command input at time k, w and v are random variables with zero mean. The EKBF operates in two stages. The first step is a dynamical update step where the state estimate and its covariances are updated according to the dynamical model of the system as:

$$\tilde{x}_k^- = f(\tilde{x}_{k-1}, u_k, 0)$$
$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$$

The super-minus symbol indicates the results are intermediate, pre-observation update, values. EKBF's work by linearizing at each time step about the state estimate, hence, the matrices A_k and W_k are the process Jacobians at step k of $f(.)$ with respect to the state estimate A and to noise W, respectively. The Jacobians must be recalculated at each time step. The matrix Q is the process noise covariance and is defined *a priori* based on knowledge of the dynamical noise components in the system.

The second stage of the filter requires incorporating observations of the real system into the state estimate. The measurement update equations are:

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T\right)^{-1}$$
$$\tilde{x}_k = \tilde{x}_k^- + K_k\left(z_k - h(\tilde{x}_k^-, 0)\right)$$
$$P_k = (1 - K_k H_k) P_k^-$$

where K_k is the so-called Kalman gain, R_k is the measurement noise covariance set *a priori*, while H_k and V_k are the measurement Jacobians with respect to the state estimate and noise, respectively. The operation of the two stages of the algorithm enable the state estimate to track the mean of the observations and the variances to represent the confidence range of the estimate.

A. Robot EKBF

To fully estimate the position and velocity of a fully holonomic robot constrained to a 2D plane, we require six state variables as indicated in Figure 2. This state representation is used for all the robot objects. Thus:

$$\tilde{x}_i = \begin{pmatrix} x_i & y_i & \theta_i & v_{//i} & v_{\perp i} & \omega_i \end{pmatrix}^T$$

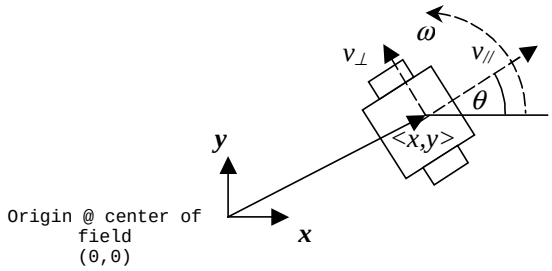

Figure 2. The six state variables and their relationship to the real world.

Given that all the field objects are represented as particles with velocity and orientation, the process update function neglecting process noise is given by:

$$\tilde{x}_k^- = B_{k-1}\tilde{x}_{k-1} + \alpha u_k \text{ with } B_j = \left(\begin{array}{c|c} \mathrm{I}_3 & R_z(\theta_j)\Delta t \\ \hline 0_3 & (1-\alpha)\mathrm{I}_3 \end{array}\right)$$

where $R_Z(.)$ is the rotation matrix given as:

$$R_z(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The velocity command is u_k, the system input, given by:

$$u_k = \begin{pmatrix} 0 & 0 & 0 & v_{//}^{cmd}(k-l) & v_{\perp}^{cmd}(k-l) & \omega^{cmd(j)}(k-l) \end{pmatrix}^T$$

here u_k is the delayed command where l is the system latency of 6 frames corresponding to a delay of about 100ms.

The dynamics update equation is developed straight from the kinematics of the system where a point particle moves with velocity $v_{//}$ parallel to its given orientation, $v_{\perp}$ perpendicular to this orientation, while rotating at ω. The α term approximately models the slop in the PID servo loops running on the robots. The α term is set to 0.5, which empirically corresponds to the average slop observed in the robots when responding to velocity commands.

Based on this difference equation, the Jacobians for the dynamics update are:

$$A = \left(\begin{array}{ccc|ccc} 1 & 0 & -\left(v_{//}\sin\theta + v_{\perp}\cos\theta\right)\Delta t & \cos\theta.\Delta t & -\sin\theta.\Delta t & 0 \\ 0 & 1 & \left(v_{//}\cos\theta - v_{\perp}\sin\theta\right)\Delta t & \sin\theta.\Delta t & \cos\theta.\Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ \hline & 0_3 & & & (1-\alpha)\mathrm{I}_3 & \end{array}\right)$$

and $W = \left(0_3 \mid \mathrm{I}_3\right)^T$

Here, $\mathbf{0_n}$ and $\mathbf{I_n}$ indicate a zero matrix and identity matrix of size *n x n*, respectively. Finally, the process noise matrix *Q*: $Q_k = diag\left(\sigma_{v_{//}}^2, \sigma_{v_{\perp}}^2, \sigma_{\omega}^2\right)$, where *diag(.)* indicates a diagonal matrix.

The standard deviations were chosen to reflect the accelerations that robots are limited to due to wheel slip. Typically robots can accelerate on the order of 3m.s^{-2} or so, thus, a working value of 4m.s^{-2} will robustly describe even the fastest robot. Working from an acceleration of 4m.s^{-2}, a robot can change its velocity by 66mm.s^{-1} in the $^1/_{60}{}^{\text{th}}$ of a second between iterations. Hence, σ_v was set to 66mm.s^{-1} for all robot types. The exception to this was the differential drive robot that had its lateral deviation set to 4mm.s^{-1} corresponding to lateral acceleration of 0.25m.s^{-2} caused by wheel slip. It was naturally assumed that opponents are capable of omnidirectional motion. For rotation, the standard deviations were set to 0.2 rad.s^{-1} corresponding to an acceleration of 4m.s^{-2} at the robot wheel. The exception is for the opponents where no orientation can be reliably observed. Hence opponents have σ_{ω} is set to zero.

Observations are only made on the positions of the objects on the field based on the vision output. Velocities are estimated purely on the observation of positions over time and the properties of the EKBF. The observation equation, neglecting measurement noise is straightforward:

$$z_k = h\left(x_k, 0\right) = \left(x_{obs} \quad y_{obs} \quad \theta_{bs}\right)^T$$

This is just the observation reported from the vision module and gives Jacobians and measurement noise as:

$H = \left(\mathrm{I}_3 \mid 0_3\right)$, $V = \mathrm{I}_3$ and $R = diag\left(\sigma_{xy}^2, \sigma_{xy}^2, \sigma_{\theta}^2\right)$

All robots are treated identically with σ_{xy} set to 25mm and σ_{θ} set to 0.173 rad (~10°) as obtained from empirical measurements of the vision output.

The observation update step is not applied blindly. False-negatives, low confidence values in the vision output, are rejected immediately and the sensing update is not performed. The dynamics update is still performed meaning the covariances on the state estimate will grow until a successful observation is reported. While this approach combats false-negatives robustly, it does not overcome the problem of false-positives, situations where the vision reports confidently, but incorrectly. False-positives, which have a disastrous impact on tracking performance, are solved with the improbability filter discussed in section IV.

The state estimate and covariance matrix need to be initialized to reasonable values to ensure their operation is sensible. Similarly, as robots can be removed or substituted during the game, it becomes necessary to reset the EKBF filter to its default, start up state. EKBF resets are driven by user input. In all reset/initialization cases the state estimate and covariance matrix are set to default values of:

$\tilde{x}_0 = \left(x^{obs}, y^{obs}, \theta^{obs}, 0, 0, 0\right)^T$ and $P_0 = diag\left(\sigma_{xy}^2, \sigma_{xy}^2, \sigma_{\theta}^2, 0, 0, 0\right)$

In short, the state estimate is set to the first observed location and orientation and the covariance matrix is initialized to the noise in location produced by sensing.

B. Ball EKBF

The ball EKBF is a little different to the robot EKBF's as it has different dynamical properties. The state vector is:

$$\tilde{x}_k = \left(x \quad y \quad v_x \quad v_y\right)^T$$

Orientation and angular velocity are dropped as they have no meaning for the ball. The ball is not driven, so there is no input command, but it has different motions depending on where it is on the field. When traveling freely along the surface of the carpeted field, the ball travels in a straight line but slows due to friction. If the ball hits an edge of the field, it rolls up and along the 45° inclined walls where friction is comparably negligible.

For normal free rolling, the primary assumption is that friction acts as a constant retarding force against the direction of motion unless the ball is traveling sufficiently slowly, where its velocity decays away quite rapidly. The model is:

$$\tilde{x}_k^- = M\tilde{x}_{k-1} + acc_k = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\tilde{x}_{k-1} + \begin{pmatrix} 1/2\, a_x \Delta t^2 \\ 1/2\, a_y \Delta t^2 \\ a_x \Delta t \\ a_y \Delta t \end{pmatrix}$$

$$a_x = \begin{cases} -a_{fr}\cos\psi & |v| > a_{fr}.\Delta t \\ -v_x / \Delta t & otherwise \end{cases}$$

Here ψ is the angle of travel of the ball, *v* is the ball speed, a_{fr} is the empirically determined acceleration constant due to friction (set to 245mm.s^{-2}). A similar expression is developed for the *y* component of acceleration. When the ball is rolling on the wall, the component of acceleration tangential to the wall is set to zero and the parallel component is set to:

$$a_x\hat{i} + a_x\hat{j} = \frac{1}{\sqrt{2}}\frac{5}{7}g\hat{n} \bullet \left(\hat{i} + \hat{j}\right)$$

This equation is developed from the dynamics of a ball rolling on a frictionless 45° wall under the influence of gravity. Here ***i*** and ***j*** are the unit axis vectors and *n* is the unit vector pointing down the slope of the wall (either ±***i*** or ±***j***).

To use the full Jacobians for these equations would result in an overly complex computation that proves unnecessary for the level of tracking required. The Jacobian becomes:

$A_k = M$ where M is specified above.

The remaining dynamics matrices are as before but are lower dimensional. They are:

$$H = (\mathrm{I}_2 \mid 0_2),\ Q_k = diag(\sigma_v^2, \sigma_v^2),\ W = (0_2 \mid \mathrm{I}_2)^T$$
$$z_k = (x^{obs}, y^{obs}),\ R = diag(\sigma_{xy}^2, \sigma_{xy}^2),\ V = \mathrm{I}_2$$

The standard deviation σ_{xy} was again chosen on empirical observations and had a value of 25mm. The velocity standard deviation σ_v was more complicated. To allow the filter to react quickly to ball-robot collisions without any explicit dynamic modeling, the state variances are increased when the ball nears a robot. This was achieved through varying σ_v depending on the proximity of the ball to a robot as:

$$\sigma_v = \max\left(\frac{R}{d}\sigma_R + \left(1 - \frac{R}{d}\right)\sigma_0, \sigma_R\right)$$

Here d is the distance between the ball and the nearest robot and R is the maximum robot width. The near robot value σ_R was set to 100mm.s^{-1}, and the open field value σ_0 was set to 10mm.s^{-1}.

C. Performance

The EKBF algorithm, described above, tracks the field objects well except when false-positives occur. With the appropriate choice of noise covariance parameters, the filter is able to compensate for the Gaussian noise components present in the sensing process but not the white noise components. Handling intermittency in the sensor data is relatively trivial by only performing the sensing update step whenever that field object is confidently identified. That is, if the confidence of identification for that particular field object is above a preset, empirically determined threshold, the sensing update step is computed. Otherwise the predict update step is the only one computed. Due to the nature of the EKBF equations, the covariance matrix P_k values will increase representing the increased uncertainty in the state estimates.

The EKBF algorithm enables one to make smart predictions about the future location of a particular field object. By repeated application of the predict update equations to the current estimate of the state vector for a particular object, one can estimate its future state and covariances enabling one to make future predictions along with their respective likelihood. Indeed, this is the approach used for overcoming the inherent latency in the vision process and by the behavior algorithms used to control the robots. Using the EKBF to predict through the inherent latency of the vision sensor substantially improves the level of achievable control with the robots, however, this is beyond the scope of this paper.

IV. Improbability Filtering

Although the EKBF works well under most conditions, it does have one flaw that severely degrades its performance. Occasionally, the vision system will misidentify a field object. On most occasions such misidentifications occur when attempting to identify robots that are partially occluded, something that occurs regularly in general play. Such false-positives in the sensing stage have a disastrous impact on the performance of the EKBF causing the state estimate to jump across the field. The impact on the robot control strategies is substantial.

The difficulty is caused by the Gaussian assumption that underlies the EKBF. When an observation differing significantly from the state estimate, as determined from the state covariances, is incorporated into the state estimate, the resulting Gaussian deviates significantly from its prior shape and position. A 1D example of this problem is shown in Figure 3. The left Gaussian represents the initial state estimate, the right Gaussian the false-positive observation. The product of the two, after normalization, gives the middle Gaussian, which is no longer a useful estimate of the true state of the system in the short term. Although the filter robustly recovers after some time, the impact of the short term deviation on the system as a whole remains significant.

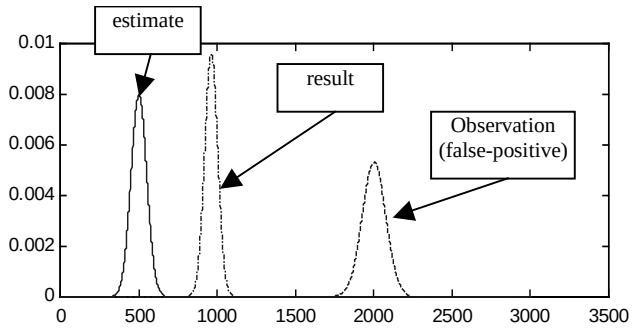

Figure 3. A 1D schematic of the effect of false-positives.

There are a number of methods that one could conceive to address the problem. Clearly, a suitably complex multi-hypothesis tracking algorithm, for example [6], could address this problem. The computational cost, however, would be prohibitive for the application discussed here. Moreover, given the frequent occurrence of false-positives, a simpler method is desirable.

More ad-hoc techniques include increasing the sensing covariances, the R matrix parameters. The drawback though is degraded performance of the EKBF under normal conditions due to its decreased sensitivity to sensor input (the K matrix has smaller values). We developed another solution that filters false-positives from observation updates by added domain knowledge. Our early investigations tried such an approach where if the observation deviated by some fixed value from the previous observation it was rejected and the filter reset. This approach however proved sensitive to parameter choice and is somewhat brittle. The question is: is there a principled way to recognize and reject false-positives that is computationally reasonable? We have devised such a method have called it *Improbability Filtering* (ImpF).

The ImpF approach is related to multi-modal EKBF in that it uses the current state and covariance estimates and the Gaussian basis of the EKBF to estimate the likelihood of observing the given observation. That is: to determine if an observation should be rejected or accepted, one needs to first calculate the conditional probability density function (pdf) $P[z_k' | x_k, P_k]$ evaluated for the given observation and state

estimates and then use this likelihood to reject observations with too low a likelihood as false-positives.

The conditional pdf is simply the Gaussian curve (in the n-dimensional space of the state vector) with the mean of the state estimate and the covariances of the state matrix where both are transformed into the observation space. Evaluating this Gaussian for the given observation will produce the likelihood of the observation with the current model. Thus:

$$P\left[z' \middle| \tilde{x}_k, P_k\right] = \frac{1}{\left(2\pi|C_k|\right)^{n/2}} e^{-\frac{1}{2}\left(z'-H_k\tilde{x}_{k_i}\right)^T C_k^{-1}\left(z'-H_k\tilde{x}_{k_i}\right)}$$

with $C_k = H_k P_k H_k^T + R$

C_k is the state covariance matrix transformed to observation space, n is the number of state variables and z' is the observation. If an observation is sufficiently likely, as determined by an empirically fixed threshold set to reject a large majority of false-positive observations, the observation is accepted and the filter is updated as per normal. Otherwise the observation is rejected and the update is skipped as in the false-negative case.

The ImpF algorithm rejects false positives in a computationally efficient manner, but does not harm the robust nature of the EKBF to incorrect state estimates. Skipped observations cause the state covariances values to increase without bound via the dynamical update step. If observations are persistently different from the state estimate, the variances will soon become large enough that any observation will become likely and the new observation will be accepted. Moreover, the large covariances will cause the state estimate to jump immediately to the new observation.

A. Experimental Results

To confirm the correct operation of the ImpF, we conducted experiments in both simulation and the real application. The results reported here focus mainly on the simulation. Results are qualitatively similar for the real application, which cannot be presented here a systematic way as false-positives cannot be generated on demand.

The simulation environment has a rich model of the robot application including the full kinematics of the system, some limited dynamics, and modeling of the latencies involved in the various components of the application. For the purposes of this discussion, it will suffice to argue that the simulator captures all the essential parts of the world bar noise that are required for tracking. Noise, an integral part of tracking problem, was modeled in two ways. Gaussian noise, drawn from a Gaussian pdf, provides the usual small signal noise observed during sensing. It accounts for any small errors introduced from vision artifacts such as pixelization, image blur, and pixel noise, among other things. False-positives, which occur due to misidentifications, were modeled as a Poisson process where the interval time between events (in multiples of frames) was drawn from a Gamma distribution. The Gamma parameter was set such that on average 1s passed between false-positive events. Intermittency was not modeled, as it is not realistic.

The following graphs show the tracking performance comparison between the raw EKBF and the ImpF + EKBF for tracking a ball while it rolls down the field and up a wall. The results for the robot tracking provide qualitatively identical and have not been shown here due to space considerations.

The three graphs in Figure 4 demonstrate the raw EKBF tracking ability in the presence of false-positives. The graphs are, in order: A) the X, Y path of the ball as tracked by the EKBF and as reported by the simulated vision. B) the X component of the EKBF and observations over time C) the distance between the real ball position and the tracked.

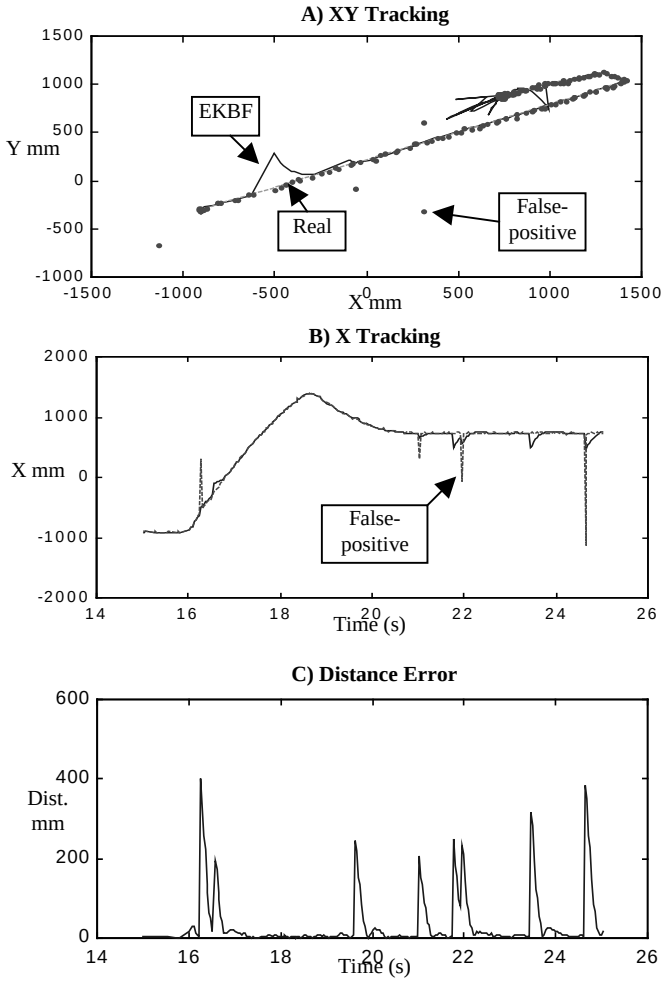

Figure 4. Performance of the raw EKBF tracking the ball hitting a wall with occasional false-positives. (A) the path of the ball, (B) the tracked position and observations, (C) the distance error between tracked position and real position.

The jagged steps in the EKBF output are clearly visible from the distance error graph. There is a one-to-one correlation between the large jumps in error and false-positives. Although the EKBF does recover quite quickly (in the order of ~250ms) these intermediate jumps significantly degrade the controllability of the system as a whole. The poor quality of the tracking is apparent by the distance error mean of 35.7mm (standard deviation of 70.0mm).

Figure 5 shows the tracking performance of the EKBF with the ImpF tracking a rolling ball hitting a wall (graph **A**) and the distance error (graph **B**). Note the different scales on the *y*-axis as compared to Figure 4, graph **C**. The error plot no longer has the large peaks corresponding to false-positives. In

comparison to above, the tracking was significantly improved with a mean error of 4.6mm (standard deviation of 2.5mm). Clearly, the ImpF performs its job as desired and greatly improves the overall robustness of the EKBF to false-positives.

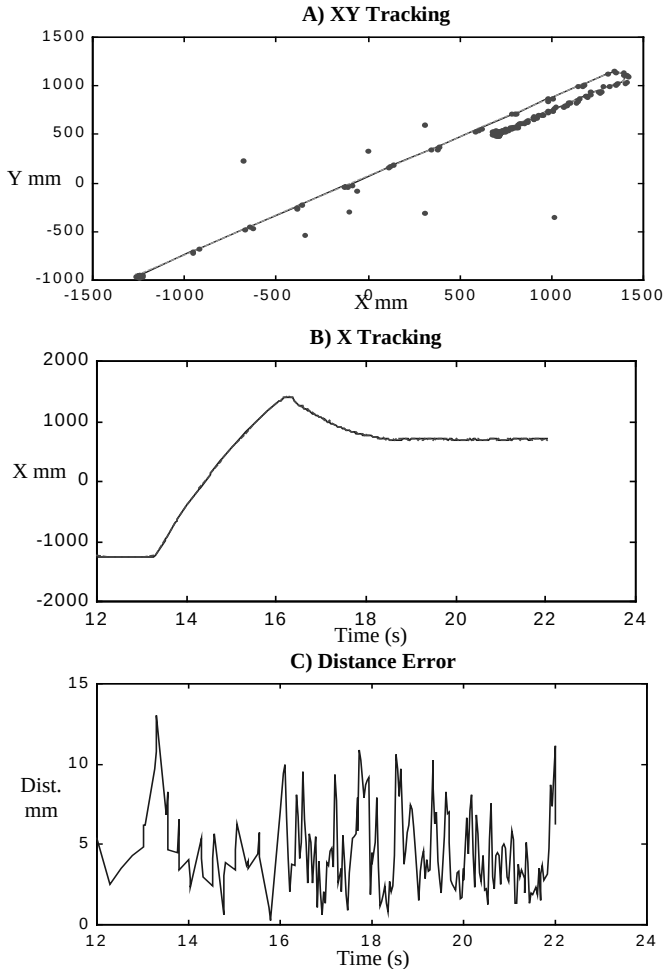

Figure 5. The ImpF + EKBF performance with (A) the path of the ball hitting a wall and (B) the distance tracking error.

V. Discussion

The experimental results verify the observed performance of the algorithm during game play, which is difficult to express quantitatively, as robust and accurate. Indeed, the system is able to operate robustly even when faced with significant occlusion and periodic misidentifications. Furthermore, its performance degrades gracefully with degraded vision input. Hence, the improbability filter performs the task it was designed for in both a principled fashion and with virtually no 'tweaking' required. The lack of 'tweaking', an attribute of robust systems more often than not, is emphasized by the rather arbitrary likelihood threshold chosen to reject false-positives. No real effort was required to tune this parameter for the level of tracking required in this application.

The above results demonstrate that in this particular application ImpF successfully removes false-positives. We conjecture that the ImpF approach is more generally applicable as a computationally light means of removing false-positives in a system that is not multi-modal in nature. Moreover, the approach rejects false-positives without introducing additional latency in observation updates or predictions and does not overly complicate the tracking module as a whole. Although applied here to an EKBF, we believe the approach may be applicable to a wider variety of probabilistic state estimators provided that an estimate of the likelihood of the observing the reported observation given the current estimated state of the system can be made.

VI. Conclusions

In this paper we have described a novel modification to the standard extended Kalman-Bucy filter approach for detecting and rejecting false-positives that would otherwise have a drastic affect on the tracking performance of the filter. We have implemented the algorithm in a working system that must track multiple robots, traveling at high speeds in a confined space, with vision as the primary sensor. Our experimental results clearly demonstrate the performance of the improbability filter approach in rejecting false-positives while not detracting from the otherwise desirable properties of the Kalman-Bucy filter. Future work must now be performed to extend this algorithm to other systems.

Acknowledgments

This research was sponsored by the United States under Grants Nos. DABT63-99-1-0013, F30602-00-2-0549, and F30602-9809135. The content of this publication does not necessarily reflect the position or the policy of the sponsors and no official endorsement should be inferred.

References

[1] Browning, B, Bowling, M., Bruce, J., Balasubramanian, R., Veloso, M. "CM Dragons 01: Team Description", *RoboCup-2001: Robot Soccer World Cup V*, Springer Verlag, submitted.

[2] Bruce, J., Balch, T., & Veloso, M. "Fast and inexpensive color image segmentation for interactive robots," *Proceedings of IROS-2000*, Japan, 2000.

[3] Chui, C. K. *Kalman Filtering: with real-time applications*, 2nd Ed., New York, Springer-Verlag, 1991.

[4] Kalman, R. E. "A New Approach to Linear Prediction and Filtering Problems", *Transactions of the ASME – Journal of Basic Engineering*, 1960, 35-45.

[5] Kalman, R. E., & Bucy, R. S. "New results in linear filter and prediction theory", J*ournal of Basic Engineering*, 1961, 95-108.

[6] Kamen, E. W., & Su, J. K. *Introduction to Optimal Estimation*, London, Springer-Verlag, 1999.

[7] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., & Matsubara, H., "RoboCup: A Challenge Problem for AI and Robotics," *RoboCup-97: Robot Soccer World Cup I*, Nagoya, L.N. on A.I., Springer Verlag, 1998, 1-19.

[8] Kwun, H, Veloso, M. "Reactive visual control of multiple non-holonomic robotic agents," *Proceedings of the International Conference on Robotics and Automation ICRA-98*, Belgium, 1998.

[9] Maybeck, P. *Stochastic Models, Estimation and Control, Vol. 1*, Academic Press Inc, 1979.

[10] RoboCup Official Website at *http://www.robocup.org/*

[11] Welch, G., & Bishop, G. "An Introduction to the Kalman Filter", *TR 95-041, Technical Report*, Department of Computer Science, University of North Carolina, NC, USA, 2001. Available from *http://www.cs.unc.edu/~welch*.