Multiple Model Kalman Filters: A Localization Technique for RoboCup Soccer

Michael J. Quinlan and Richard H. Middleton

[1] Department of Computer Science, University of Texas at Austin, USA
[2] Hamilton Institute, NUI Maynooth, Maynooth, Co. Kildare, Ireland
mquinlan@cs.utexas.edu, richard.middleton@nuim.ie

Abstract. In the Standard Platform League (SPL) there are substantial sensor limitations due to the rapid motion of the camera, the limited field of view of the camera, and the limited number of unique landmarks. These limitations place high demands on the performance and robustness of localization algorithms. Most of the localization algorithms implemented in RoboCup fall broadly into the class of particle based filters or Kalman type filters including Extended and Unscented variants. Particle Filters are explicitly multi-modal and therefore deal readily with ambiguous sensor data. In this paper, we discuss multiple-model Kalman filters that also are explicitly multi-modal. Motivated by the RoboCup SPL, we show how they can be used despite the highly multi-modal nature of sensed data and give a brief comparison with a particle filter based approach to localization.

1 Introduction

Localization has been studied by many researchers for several years now. Most of the algorithms implemented in RoboCup fall broadly into the class of particle based filters (see for example [5]) or Kalman type filters (see for example [6]) including Extended and Unscented ([4]) variants. In some divisions of RoboCup, algorithms are very well established, given the rich sensor data provided by laser scanners, omni-directional cameras etc. (see for example [3]). However, in the standard platform league (formerly the four legged league) there are substantial sensor limitations particularly with the rapid motion of the camera, and the need for active perception. In addition, the league has deliberately removed beacons as unique landmarks, leaving the colored goals as the only unique landmarks on the field.

Due to the non-uniqueness of most land marks in the SPL, it is important that any localization algorithm be able to handle this ambiguous data. In particular, it is clear that in many cases, the relevant probability density functions will be multi-modal. Whilst it is generally accepted that particle filters can handle this situation, it seems less well known in the RoboCup domain that Kalman type filters can be easily adapted to handle multi-modal distributions. In other research areas, however, multiple model (also called Gaussian Mixture or Gaussian Sum) filters have been used for many decades (see for example [1]).

In this paper, we first give a review of multiple-model Kalman filters, with particular emphasis on features and approximations relevant to real time implementation within the RoboCup framework. We then present examples and results of the mutliple model Kalman filter.

2 Multiple Model Kalman Filters

2.1 Problem Formulation

In many robotics applications, localization algorithms are concerned with the problem of estimating the 'state' of the robot, from uncertain data. For example, in the Standard Platform League, we might typically be concerned with estimating the location (in 2D cartesian coordinates) and orientation of the robot given data derived from vision of objects on the field such as goal posts and field markings. In this case, the state we wish to estimate is written as the 3 dimensional vector

$$x(t) = \begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta(t) \end{bmatrix} \tag{1}$$

where (x_r, y_r) denote the robot's cartesian coordinates and θ is the robot's orientation. Often, a probabilistic or statistical representation of uncertainty is used, though more recently some 'constraint' based localization techniques have also been applied (see for example [2]).

In the probabilistic setting, adopting a Bayesian estimation framework, there are two components to the state estimation problem:

- **Time Update.** Firstly, given the pdf of $x(t-1)$ conditioned on data up to time $(t-1)$, $p(x(t-1))$ and also given odometry information at time t, we wish to make an estimate of the conditional density function of $x(t)$, given data up to time t, $p(x^-(t))$.
- **Measurement Update.** Secondly, given the conditional pdf, $p(x^-(t))$and also given measurement data at time t, we wish to find $p(x(t))$.

In the standard Kalman filter approach, we use a multivariate (n-dimensional) Gaussian to represent the conditional pdfs of $x(t)$, for example

$$p(x(t)) = \frac{1}{(2\pi)^{n/2}|P(t)|^{1/2}} e^{-\frac{1}{2}(x(t)-\hat{x}(t))^T P^{-1}(t)(x(t)-\hat{x}(t))} \tag{2}$$

where $\hat{x}(t)$ denotes the expected value of the state at time t, and $P(t)$ represents the state covariance matrix also at time t. In the standard Kalman filter, or the extended or unscented versions there are simple expressions that allow computation of the time update equations (that relate $(\hat{x}^-(t), P^-(t))$ to $(\hat{x}(t-1), P(t-1))$ and measurement update equations that relate $(\hat{x}(t), P(t))$ to $(\hat{x}^-(t), P^-(t))$).

The Kalman filter has an extensive history and has proven very useful in a wide range of applications, and also enjoys relatively simple computations. At

each time, given a scalar measurement variable, the computational complexity of the time update and measurement update equations is typically $O(n^2)$ where n is the dimension of the state variable. Unfortunately, it provides a very poor representation of multi-modal distributions, since despite the generality available in (2), this distribution is always unimodal. Fortunately, this difficulty can be overcome by the use of Gaussian mixtures.

2.2 Gaussian Mixture background

Gaussian mixtures represent the state pdf as a sum of a number of individual multivariate Gaussians, or multiple models. Each of the N models, for $i = 1..N$, is described by 3 parameters (where for simplicity we drop the explicit dependence on time):

- $\alpha_i \in [0, 1]$, the probability that model i is correct, that is, the 'weight' associated with model i,
- $\hat{x}_i \in \mathbb{R}^n$, the state estimate for model i,
- $P_i = P_i^T > 0 \in \mathbb{R}^{n \times n}$, the covariance for model i.

For each model, the multivariate normal probability distribution function (pdf) is given by:

$$p_i(x) = \alpha_i \frac{1}{(2\pi)^{n/2}|P_i|^{1/2}} e^{\left(-\frac{1}{2}(x-\hat{x}_i)^T P_i^{-1}(x-\hat{x}_i)\right)}. \tag{3}$$

The overall mixture pdf is therefore:

$$p(x) = \sum_{i=1}^{N} p_i(x). \tag{4}$$

Note that all variables, α_i, $\hat{x}_i$, P_i and N can change with time in the algorithms to be discussed below.

Some of the key features of this representation are that under certain assumptions, any pdf can be approximated to an arbitrary degree of accuracy by a Gaussian mixture of sufficient degree (see for example the discussion in [1, §II]). We first consider the simple case of updates for unambiguous measurements.

2.3 Model update with unambiguous measurements

We first perform an EKF (or UKF as appropriate) update of each of the N models, for all unambiguous objects from vision (e.g. ball, known goal posts, field markings that can be uniquely identified from other visual cues). This EKF (or UKF) measurement update is identical to the regular (that is single model) update, except that we need to include update equations for the model weight. For each model, and for each measurement update, an approximate heuristic for updating the weights is:

$$\alpha_i := \alpha_i \left(\frac{R}{R + (y - \hat{y})^2} \right) \tag{5}$$

where R is the variance of the measurement considered. Note the update proposed in (5) is simple and has the right general form, that is, zero innovation keeps the α value high, whilst large innovation shrinks the value. However, assuming statistically independent normally distributed measurement errors, and allowing for vectors of m measurements[3] the weights should be updated according to:

$$\alpha_i := \alpha_i \left(\frac{1}{\sqrt{(2\pi)^m |\Sigma_\eta|}} e^{-\frac{1}{2}\eta^T (\Sigma_\eta)^{-1} \eta} \right) \tag{6}$$

where $\eta = y - \hat{y}$ is the innovations associated with the measurement, and Σ_η is the variance of the innovations. Note that the innovations variance can be computed as the sum of the measurement variance R and the variance $\hat{R}$ of the prediction, $\hat{y}$, that is, $\Sigma_\eta = R + \hat{R}$.

One of the problems with the weight update given in (6) is its lack of robustness to outliers. For example, a single, slightly bad measurement where $|\eta| = 4\sigma_\eta$ would multiply α_i by almost four orders of magnitude less then if $\eta \approx 0$. To correct this, if we assume a probability of ϵ_o that our observation is an outlier (that is a false positive from vision), then a more appropriate weight update is:

$$\alpha_i := \alpha_i \left((1 - \epsilon_o) \frac{1}{\sqrt{(2\pi)^m |\Sigma_\eta|}} e^{-\frac{1}{2}\eta^T (\Sigma_\eta)^{-1} \eta} + \epsilon_o \right). \tag{7}$$

Having processed all the unambiguous measurements, we now turn to the problem of processing ambiguous measurements, which gives rise to the problem of model splitting.

2.4 Model Splitting - ambiguous measurements

When considering an ambiguous measurement update, with M alternate possibilities, an initial distribution with N elements (or models) can be performed by splitting each of the N initial models into M models (to a total of $M \times N$ models) and doing a standard measurement update for each possible combination of model component with each possible measurement component. Note that it is also possible that splitting could be the results of ambiguous time updates (for example, if we are uncertain whether the ball has just been kicked). In this case, similar considerations to those below will apply during the time update portion of the extended Kalman filter. For now, we look just at the measurement update equations.

Suppose that we start with N models, and a measurement that is ambiguous, and can therefore be interpreted as M different field objects, such as M different corner points. For simplicity we consider the case where each of these is equiprobable, though there is no difficulty in generalizing the algorithms below

[3] For example, it may make sense to consider the range and bearing of a single object as a single, two dimensional vector measurement.

to cases where each of the measurement ambiguities has different, but known, probabilities.

The processing of an ambiguous measurement is performed by executing the following actions:

```
for each active model i
   for j =1 to number of ambiguous choices
      create a new copy (child) of model i;
      update this new model with measurement type j;
      if (update is an outlier) merge[4] new model with model i;
   end;
   renormalize the weights for all children of model i;
end;
```

Note that the distribution of weights at the end of the inner loop respects the relative weights after the measurement updates, but renormalizes the total weight. Clearly, whilst the individual actions within this procedure are relatively computationally cheap, it can give rise to an exponential growth in the number of active models, which is clearly impractical. One of the most important problems therefore in many multiple model Kalman filters is how to control the number of models. Although pruning (that is deleting) models with very small weights may be helpful, this is not a complete solution and it is important to have procedures for merging models.

2.5 Model Merge Equations

We first consider the simpler of the problems associated with merging models, namely, given a group (often a pair) of models, how do we merge (or join) these into a single resulting model that approximates the original pdf. There are many possible algorithms that may be used for merging models, see for example [8]. The discussions here follow closely these algorithms or simplified forms of them. For simplicity, we describe merging a pair of models, however, the algorithms below generalize trivially to merging more than two models at once.

Firstly, it is clear that when merging, to preserve the total weight probability of one of the models being correct, we should have [8, (2.24)]:

$$\alpha_m = \alpha_1 + \alpha_2 \tag{8}$$

where α_m is the weight of the merged model and α_1, α_2 are the weights of the two models to be merged.

[4] The model merge procedure will be discussed in Section 2.5. This logic frequently causes early model merges and thereby reduces the expansion in the number of active models.

Also, we can derive the merged mean as follows[5]:

$$\hat{x}_m = \frac{1}{\alpha_m}\left(\alpha_1 \hat{x}_1 + \alpha_2 \hat{x}_2\right) \tag{9}$$

Note however that this merging algorithm can cause 'drift', wherein, merging of a high weight, though slightly uncertain model, and low probability model with different mean, causes a small shift in the parameter estimates. If this situation persists (for example when repeatedly observing the same ambiguous object without any other observations), then the parameter estimates can drift significantly. To avoid this problem prior to computing the merged covariance, we follow (9) by the logic:

$$\text{if } \alpha_1 > 10 * \alpha_2 \textit{ then } \hat{x}_m := \hat{x}_1$$
$$\text{if } \alpha_2 > 10 * \alpha_1 \textit{ then } \hat{x}_m := \hat{x}_2$$

If we wish to preserve the overall covariance of the pdf corresponding to the original pair of pdfs, then the merged covariance matrix is given by:

$$P_m = \frac{\alpha_1}{\alpha_m}\left(P_1 + (\hat{x}_1 - \hat{x}_m)(\hat{x}_1 - \hat{x}_m)^T\right) + \frac{\alpha_2}{\alpha_m}\left(P_2 + (\hat{x}_2 - \hat{x}_m)(\hat{x}_2 - \hat{x}_m)^T\right) \tag{10}$$

Note that it is not obvious that these equations give the 'optimal' merge. In particular, some of the main contribution of the thesis [8], is to pose the merge problem as an optimization of the difference between the resultant pdf and the original mixture pdf. In this case, a recursive algorithm for computing the optimal merge can be generated. For reasons of simplicity and numerical efficiency, we propose the simpler equations (8),(9),(10). Note however, that (for example) merging a low weight high variance pdf with a high weight low variance pdf by this procedure tends to under-estimate the probability of the 'tail' of the distribution, whilst giving better accuracy in the pdf of the main mode of the distribution.

2.6 Model Merge Decisions

Model merge decisions are complex and there seem to be a number of possible algorithms for this. The authors of [8] formulate the problem of deciding which models to merge in an optimization framework. This optimization starts with a high order mixture model and seeks to find a lower order mixture model that best fits the original mixture model in the sense of the mean square deviation of the probability density functions. The only inputs needed are the original model, and the number of elements (models) in the final mixture. However, the computations for this kind of procedure seem prohibitive in the RoboCup SPL environment.

[5] Note that when merging, extra care is need to merge the orientation components of the estimates. For example, merging an orientation of 179° with -179° should not give 0°.

We therefore propose a computationally simpler procedure, based on a simplified form of the optimization approach. Our approach is based on computing pairwise merge metrics, that is, a measure of how much 'information' will be lost if this pair of models is merged. One metric proposed for example in [8, pp2.66] computes the distance metric, d_{ij}, for a pair of models indexed by (i, j) as

$$d_{ij} = \left(\frac{\alpha_i \alpha_j}{\alpha_i + \alpha_j}\right)(\hat{x}_i - \hat{x}_j)^T P_{ij}^{-1}(\hat{x}_i - \hat{x}_j) \tag{11}$$

where P_E is the covariance that would result if the models were to be joined,

$$P_{ij} = \left(\frac{\alpha_i}{\alpha_i + \alpha_j}\right) P_i + \left(\frac{\alpha_j}{\alpha_i + \alpha_j}\right) P_j + \left(\frac{2\alpha_i \alpha_j}{\alpha_i + \alpha_j}\right)\left((\hat{x}_i - \hat{x}_j)(\hat{x}_i - \hat{x}_j)^T\right) \tag{12}$$

To simplify calculations, and avoid the matrix inverse that may be problematic in higher order systems (for example combined robot, ball and ball velocity estimation where $n = 7$), we propose a simpler approximate metric

$$D_{ij} = (\alpha_i \alpha_j)\,(\hat{x}_i - \hat{x}_j)^T\,(\alpha_i \Delta_i + \alpha_j \Delta_j)^{-1}\,(\hat{x}_i - \hat{x}_j) \tag{13}$$

where Δ_i denotes the matrix formed by the diagonal component of P_i.

Given a mechanism, such as (13), for computing a metric on the closeness of two models, we could repeatedly search for the closest two models to merge. Note however, that to implement this, we must first compute all $N(N-1)/2$ possible distance metrics, find the smallest, merge these, then recompute merge metrics (or at least the $N-2$ metrics associated with the new merged model) and repeat. Alternatively, it may be simpler to merge based on a threshold using an algorithm such as that following:

```
for each active model i
   for each active model j
      if (mergeMetric(i,j) < threshold) then
         model i := mergeTwoModels(i,j);
   end;
end;
```

If this threshold based merge does not achieve sufficient reduction in the number of models, it can be repeated with larger thresholds.

2.7 Algorithm Implementation

To avoid the overhead of dynamic memory allocation and deletion, (as well as the potential to inadvertently create memory leaks), we implement a fixed size array of size `MAX_MODELS` of models. Each of these models is a normal KF model (that is, includes the states estimates and the state covariance) and in addition includes parameters for the weight `alpha` and a Boolean, `active`, denoting whether or not the model is in use.

The execution has main steps as follows:

1. **Time Update** For each of the active models, a call is made to the regular KF time update on this model, that is, it incorporates locomotion data and updates the filter covariances.
2. **Measurement Update for all Unambiguous Objects** For each of the active models, a regular measurement update is performed as suggested in Section 2.3.
3. **For each Ambiguous Object, split models**. Ambiguous objects are: (i) Unknown Intersections; (ii) Unknown Lines; and (iii) Ambiguous Goal Posts. For each of these situations, each active model is split according to the various possibilities for the unknown object. This splitting uses algorithms as discussed in Section 2.5. After this process, the model weights, α_i, are normalized so that they sum to 1.
4. **Merge Models**. Since splitting models can leave us with a large number of possible models, after each ambiguous object update, we merge models to try to eliminate redundant models.
5. **Generate Localization Data for Behavior** Following the model merge, the α_i values are again normalized and the best model is selected to represent the most likely robot position, together with variances of the estimates. Note however, that we have a special segment of code, so that if there is a valid 2nd best model, the variance of the estimates reported to behavior is increased to account for any deviation between the state estimates for the best and 2nd best models. For example, with respect to orientation, instead of reporting just the variance $\sigma^2_{\theta i_1}$ of the best model i_1, the overall heading variance σ^2_θ is computed as
$$\sigma^2_\theta = \sigma^2_{\theta_{i_1}} + \alpha_{i_2} \left(\theta_{i_1} - \theta_{i_2}\right)^2 \tag{14}$$
where i_2 denotes the index of the second most likely model.

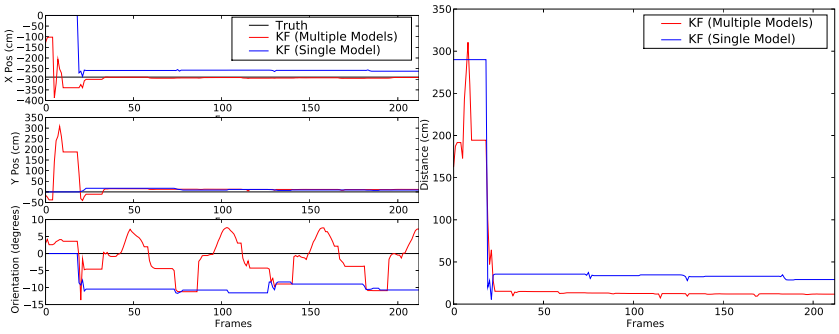
3 Example and Results

In this section we run through an example of a Multiple Model Extended Kalman Filter (MM-EKF), as described in Section 2. One of the most demanding localization situations in RoboCup SPL is goal keeper localization since positioning needs to be very accurate, and in most cases, the only visible unique land marks are distant goals. In our test case we have a robot standing inside the yellow goal mouth looking directly up the field (in our coordinate system, the location is x=-290, y=0.0, θ=0.0). The robot then pans its head from side-to-side. In this example, the robot saw 46 unique observations (either the blue goal or identifiable blue goal posts) and 278 ambiguous observations (unidentifiable blue goal posts, intersections and lines). This gives an indication of the amount of information being ignored when not using ambiguous objects.

3.1 Comparison with a Single Model EKF

Firstly, lets present the accuracy results when comparing a MM-EKF to a Single Model EKF (S-EKF). As expected the MM-EKF easily out performs the S-EKF

(see Figure 1). Note: the error at the start is due to the robot not intially knowing its location, once settled both versions converge to a stable location. In this case, the MM-EKF converged to a location 11.61cm from the true location with an average orientation error of $-1.6°$. While the S-EKF converged to a location 29.12cm from the real location and with an average orientation error of $-9.30°$.



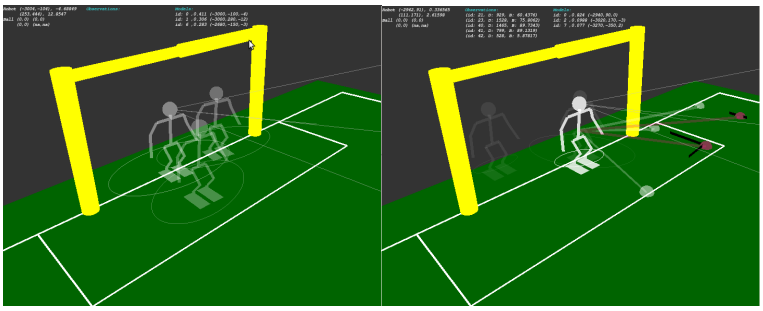
(a) (b)

Fig. 1: Accuracy comparison between Multiple Model EKF and Single Model EKF. (a) Presents the location from the filters in each of x, y and orientation (θ). (b) Total distance error from actual location.

3.2 Splitting and Merging

We will present two examples of splitting and merging. Firstly, an example of a merge split/merge cycle where three roughly equal models can converge to a more likely model after observering only ambiguous information. This example describes the update taking place at Frame 33 from Figure 1. Originally the MM-EKF is maintaining three models (as shown in Figure 2 (a)) with α values of 0.411,0.306 and 0.283 respectively. The opacity of the robot represents the α value of that model, with a more solid robot representing a higher α.

The first observation considered is the intersection that is 93cm away at an angle of 60°, in the ideal case this observation should be matched with the top left corner of the penalty box. Unfortunately the three models have just enough uncertainty that they keep the 3 corners to the left of the robot, that is the top and bottom corners of the penalty box and the corner on the left hand side of the field. This spilt can be seen in Figure 3 (a), with the corresponding merge (d) reducing the total down to two models. Next the MM-EKF considers the second (false) intersection. Again the same three corners are considered valid options and the proceeding split/merge trees are shown in (b) and (e) respectively (Note: Model 6 is still alive at the end of the merge but was not graphed due to nothing combining with it).



(a) Before Observations (b) After Observations

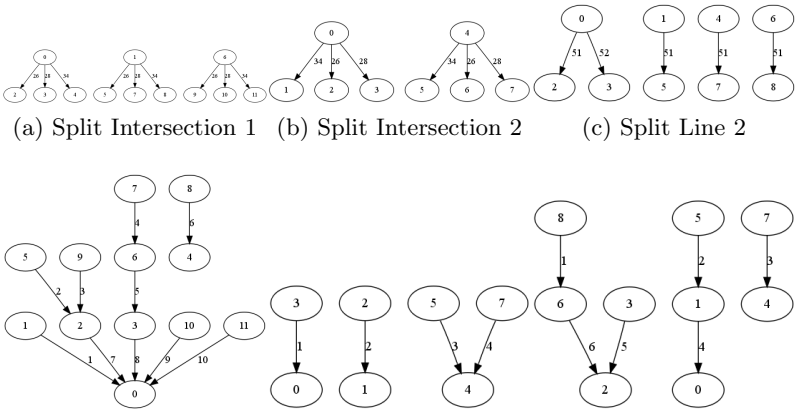
Fig. 2: Example of ambiguous observations converging to the correct location. (a) 3 almost equal probability models (α=0.411,0.306,0.283). (b) After the observation of 3 lines and one intersection we are now left with a high certainty model (α=0.824) and two lower likelihood models (α=0.099,0.077).

The next observation considered is the line to the far left. This should be the left sideline but its has been reported with an incorrect vision distance. Because of this error all the alternatives we rejected as outliers, hence no splits or merges we undertaken. The next considered observation is the left edge of the penalty box, in this case three of the four models (1,4 & 6) outliered on all but the correct line, while model 0 was sufficiently uncertain that it also considered the sideline as an alternative (shown in (c)). For space reasons we have forsaken showing the splits on the front edge of the penalty box, but rather have shown the final merge tree of the combined splits for the last two observed lines (f).

Secondly, we show the worst case scenario, that is when the robot has very little idea where it is (i.e. a high variance in location) and it observes an ambiguous object (Figure 4). In this case the MM-EKF is maintaining 3 possible models and then the robot observes an unknown intersection. The models fail to outlier on most of the possible alternatives and the merging step runs through a complicated procedure as shown in Figure 4 (b). The end results show very little improvement in accuracy as no alternative is favored. Luckily this scenario only occurs when the robot is kidnapped (or at startup) and doesn't see many unique objects before seeing an ambiguous object.

3.3 Comparison with a Particle Filter

Here we briefly compare the the MM-EKF to a Particle Filter (PF). The PF used in this comparison has been run at both the 2007 and 2008 RoboCup competitions and has been tuned for game performance. While the both filters provide similar accuracy, the MM-EKF can do so with a high certainty. This is due to the PF using a small number of particles (100) to simultaneous track position and handle noise/outliers/kidnapping. While the MM-EKF can perform optimal updates based on the observations for each model, while relying on more



(a) Split Intersection 1 (b) Split Intersection 2 (c) Split Line 2

(d) Merge Intersection 1 (e) Merge Intersection 2 (f) Merge Lines 2 & 3

Fig. 3: (a)-(f) present the sequence of splits and merges that take place due to the observations.

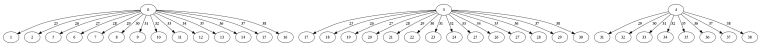
sophisticated approaches to outliers and kidnapping. It should also be noted that even when running so few particles the MM-EKF is substantially faster in terms of execution time, averaging less then 35% of time required to process a vision frame.

4 Conclusions

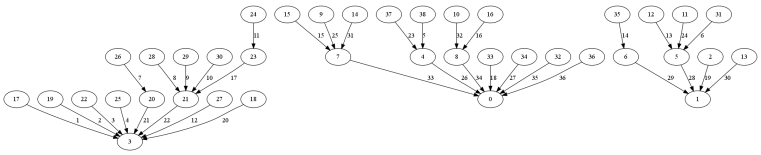
In this paper, we have discussed the background and implementation of multiple model Kalman filter based localization, with particular emphasis on the RoboCup Soccer Standard Platform League. The MM-EKF is able to directly handle the ambiguous information, and therefore resultant multi-modal distributions common in the SPL. It shows performance that is substantially better than standard EKF implementations, and at least in a preliminary test, outperforms a particle filter applied to the same problem. The main complexity with the MM-EKF is the merge decisions required to keep the number of active models limited to a fairly low number. We have given some simple algorithms designed to achieve this with low average, and moderate peak CPU demands. Further work on a more detailed comparison with a wider variety of particle filters is required to give a more accurate picture of the relative merits of the different approaches.

References

1. D. Alspach and H. Sorenson, “Nonlinear Bayesian estimation using Gaussian sum approximations”, IEEE Transactions on Automatic Control, V17, N4, pp439-448, 1972.

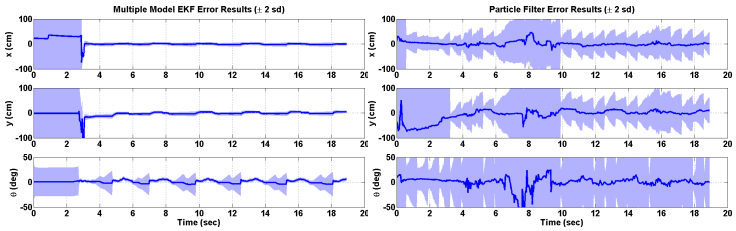


(a) Split when models are uncertain



(b) Merge after uncertain split

Fig. 4: Example of the worst case splitting/merging that can occur when seeing an ambiguous object. In this case the robot was unsure of its own location and observed an intersection. Unable to outlier many of the alternatives forced it perform a complicated merge.



(a) (b)

Fig. 5: Comparison of the MM-EKF (a) and a Particle Filter (b). The shaded area represents the uncertainty (two standard deviations).

2. D. Goehring, H. Mellmann and H.-D. Burkhard “Constraint Based Object State Modeling.” In: European Robotics Symposium, Lecture Notes in Artificial Intelligence. Springer 2008.
3. J.-S. Gutmann, T. Weigel and B. Nebel, “Fast Accurate and Robust Self Localization in the RoboCup Environment”, Proc 1999 RoboCup Symposium, Stockholm.
4. S. Julier, J. Uhlmann, and H.F. Durrant-Whyte, “A new method for the nonlinear transformation of means and covariances in filters and estimators”, IEEE Transactions on Automatic Control, V45, N3, pp477-482, 2000.
5. S. Lenser and M. Velosa, “Sensor resetting localization for poorly modeled mobile robots”, Proc. of ICRA-2000, San Francisco, April 2000.
6. R.H. Middleton, M. Freeston and L. McNeill, ”An Application of the Extended Kalman Filter to Robot Soccer Localisation and World Modelling”, Proc. IFAC Symposium on Mechatronic Systems, Sydney, September, 2004.
7. W. Nistico and M. Hebbel, “Temporal Smoothing Particle Filter for Vision Based Autonomous Mobile Robot Localization”, Proc. 5th International Conference on Informatics in Control, Automation and Robotics, Funchal, May, 2008.
8. J.L. Williams, “Gaussian Mixture Reduction For Tracking Multiple Maneuvering Targets In Clutter”, PhD Thesis, AFIT/GE/ENG/03-19, Wright-Patterson Air Force Base, Ohio, 2003.