Switching Between Extended Kalman Filter and Particle Filter – Project Report in Applied Estimation

Hugo Blanc
880912-2875

Sebastian van de Hoef
871204-4232

January 8, 2012

Abstract

In this report we present the results of the project assignment we conducted for the course "Applied estimation" at KTH, Stockholm. A particle filter and an extended Kalman filter for robot localization investigated in the preceding laboratory exercises were slightly modified and combined into one estimator. Based on the estimated variance of the belief the filter switches between the particle filter and the extended Kalman filter. A number of simulations were conducted and the results are presented in this report. The simulations indicate that in some situations the concept can overcome the limitations of the extended Kalman filter while being more computationally efficient than the particle filter.

1 Introduction

In many control applications it is required to know the full state vector of a dynamical system online. However it might not always be possible to directly measure all states but only indirectly. Furthermore the measurements might be corrupted by noise. In this case it is necessary to estimate the state in some way. This is frequently done by recursive Bayesian state estimation [8].

There are a variety of different implementations of this rather abstract concept. For nonlinear dynamical systems the extended Kalman filter (EKF) is extremely popular [8]. The EKF is easy to implement and computationally efficient. However it is a well known problem that EKF diverges in some situations [7]. In [5] the author elaborates in detail how this divergence can happen. Furthermore the EKF assumes a Gaussian distribution of the estimation, which makes it unsuitable if there are multiple hypotheses or no initial information of the state.

Due to these constraints the particle filter (PF) has become very famous in robot localization the recent years [8]. It is a non-parametric filter, which represents the state estimate by finite set of samples (the particles). Hence it is not restricted to any predefined shape of the estimate distribution. Another advantage is, that it is easy to incorporate constraints into the estimation. However these advantages come with the price of high computational effort and discretization errors.

There are a variety of variations on both filters to address these problems. For multiple hypotheses due to ambiguous data association the multi-hypotheses tracking filter can for instance be employed [8]. This filter suffers however also from the divergence issues described in [5].

The idea of this project is to switch between the EKF and the PF. In situations where the EKF works well it should be used to create precise estimates with low computational effort. This could either save electrical power or the unused computational power could be used for other tasks, for instance to process more measurements or to run the filter at higher frequency. The latter points were not investigated in this project.

In situations in which the EKF would fail the system should automatically switch to the PF. Then the PF can bridge the time until it's again possible to use the EKF.

It was quite surprising to us how little work on this topic could be found. In [4] the same idea is used for color tracking switching between a linear Kalman filter and a particle filter. Here the Kalman filter is used to track fast moving objects. As soon as the objects start changing direction rapidly they switch to a particle filter.

In [6] a PF is used to initialize a bearing only SLAM to estimate the relative pose of two robots. As soon as the particle distribution resembles a Gaussian they switch to an Unscented Kalman filter (UKF). Furthermore they test the innovation to detect divergence of the UKF. If the normalized squared innovation is greater than a threshold the filter is re-initialized with the particle filter.

In [1] the PF is also used to initialize an UKF. Based on the pitch response a car was localized on a test track. In the beginning the PF was used for global localization. A test similar as in [6] was applied to determine the switchover point to the UKF. The threshold for the switching is systematically chosen as a trade-off between computational effort and estimation accuracy. A normalized innovation squared test is applied to re-initialize the estimator from the beginning.

2 The Estimation Task

In this project we looked at robot localization based on range-bearing measurements to point features with unknown associations and with a given map and odometry data. This is exactly the same estimation task as in lab 1 and 2. Please find the description of the estimation task in [2] section 3.2.

3 Particle Filter

The particle filter is basically the same as we used in the lab 2 with systematic re-sampling. We actually reused the Matlab code written in the laboratory exercise. You can find the detailed description of the filter in [3].

However we made two changes to the version used in the lab.

Firstly the filter needed to be able to run without any measurement in certain time slots. If there is no measurement only the prediction for each particle is carried out. The whole process of assigning weights and re-sampling is skipped in that case.

Furthermore we added the ability to consider constraints in the form of rectangular walls. The walls are defined in a text file where each line defines a rectangle in which the robot is considered not to be able to enter. The structure is lower left corner x and y and then upper right corner x and y separated by white spaces.

If the parameter `MIND_WALLS` in `mcl.m` is set to 1 then the weights of all particles which are at least in one of the rectangles specified in the text file mentioned above are assigned a weight of zero. Then the weights are normalized to a sum of 1. In the subsequent systematic re-sampling step these particles are removed due to their zero weights.

In the unlikely situation where all particles are found to be in a wall then a new set of particles is created with equal weights and uniformly scattered in the space $[-\pi, \pi]$ for the orientation and in the space with less the distance defined in the parameter `bound_redis` in the file `mind_walls.m` from the mean of the previous particle set for the position.

Another nice property of the PF is that it requires quite little computational power if there are no or few measurements. In the case of no measurements only the prediction step has to be carried out. These are exactly the situations when the uncertainty is large and the PF is used in our estimator.

To account for the discretization error of the particle filter it has proven to be useful to model the measurement noise much higher than only the true measurement noise. Otherwise multiple hypotheses get easily dismissed as the particle set covering one hypothesis by chance explains the measurements better as the other. Furthermore there is the chance that if the particle set is spread out after a long period without measurements the variance of the particle set decreases too quickly before the PF managed to re-concentrate the particle set around the true state and hence the EKF gets initialized too far off the true pose. As the noise model for the EKF is different the loss in accuracy due to high values for the compensation noise is not really a problem.

4 Extended Kalman Filter

For the extended Kalman filter we also reused the code from lab 1. See [2] for details. As for the particle filter we also had to adapt the code so it would run with no measurements. In this case only the predict step is executed and not the update step.

5 Switching Between the Two Filters

The switching between the two filters is the critical point in this project. The EKF can only run, when the belief distribution is close to a Gaussian and it's mean close to the true systems state. Otherwise the linearization or the data association might be invalid.

[1] indicates, that it's not sufficient to only test the goodness of fit of the belief to a Gaussian, as proposed in [6]. They suggest comparing the product of the misfit and the variance of the belief distribution to a threshold.

For the estimation problem at hand experience from the particle filter indicates that non-Gaussian distributions occur mostly if there is huge uncertainty in the belief. Then we also have the problem that the mean of the pose estimation might be far of from the real pose. Therefore we suggest considering only the weighted sum of the belief variance for the switching.

With $\sigma_x^2, \sigma_y^2, \sigma_\theta^2$ as the diagonal elements of the belief covariance matrix Σ we have the reference value:

$$\sigma_{\text{ref}}^2 = \frac{a^2\sigma_x^2 + a^2\sigma_y^2 + \sigma_\theta^2}{2a^2 + 1} \tag{1}$$

where a is parameter to scale between the angle and the position.

5.1 Choice of the right starting filter

If the start pose of the robot is well known, the localization task starts with the EKF. In that case uncertainty is small and the EKF can linearize close to the true state of the system. In those cases the EKF tends to work well.

If we don't have any knowledge (or a very poor knowledge) of the initial pose we start by performing a global localization problem using the particle filter similar as in [1]. We begin with spreading particles randomly and uniformly over the considered state space. That is $[-\pi, \pi]$ for the orientation and rectangular area of the map. So it allows us to track multiple hypotheses (e.g. in symmetrical environments) assuming that the parameters and the number of particles of the filter are picked such as particle deprivation does not happen too quickly, e.g. the filter collected sufficient data to dismiss all but one hypothesis.

5.2 Switching from PF to EKF

For the switch from PF to EKF we compare σ^2_{ref} to a threshold $\sigma^2_{\text{PF to EKF}}$. If we have $\sigma^2_{\text{ref}} < \sigma^2_{\text{PF to EKF}}$ then the filter switches to the PF. Furthermore we check that not more than a certain amount of particles (t_{pc}) violated the constraints in the preceding update. This is, because the EKF cannot deal with constraints as well was the PF does. Furthermore this might cause the problem that the filter switches to the PF. But as the PF considers the constraints, the variance of the particle set gets small enough to switch back to the EKF so that the filter switches all the time.

The initial conditions for the EKF are estimated from the particle set taking the mean and the variance of the particle set.

5.3 Switching from EKF to PF

For the switch from EKF to PF we compare σ^2_{ref} to a threshold $\sigma^2_{\text{EKF to PF}}$. If we have $\sigma^2_{\text{ref}} > \sigma^2_{\text{EKF to PF}}$ then the filter switches to the EKF. The initial particles are drawn from the belief distribution of the EKF, which is a Gaussian.

Furthermore we also test the health of the filter as done in [1]. We use the same χ^2-test as for the outlier detection. However we compare the mean of the Mahalanobis distances for all the measurements at a time step. If the test fails in a number of consecutive time steps a global localization with the particle filter is triggered. We take the mean and require the test to fail $t_{\text{reloc\_window}}$ times to prevent an outlier to trigger a global localization. With that mechanism the estimator can also handle the kidnapped robot problem, if the measurements are not too ambiguous.

There is a number of other approaches to monitor the health of the EKF. One could for instance monitor if the predicted mean violates constraints, if there are sudden changes in the measurements, negative data or data from other sources.

5.4 Choice of the Thresholds

It is clear that $\sigma^2_{\text{EKF to PF}}$ may not be much smaller than $\sigma^2_{\text{PF to EKF}}$ otherwise the filter might switch back and forth all the time.

For $\sigma^2_{\text{EKF to PF}}$ the easiest way to determine the threshold is as done in [1] to run first only the particle filter and choose a value with some margin above the values of σ^2_{ref} when the filter converged in places with many measurements, where we want the filter to switch to the EKF. This can be done by setting a parameter in the file `init.m`. For our test cases σ^2_{ref} drops significantly as soon as the PF gets good measurements.

For $\sigma^2_{\text{PF to EKF}}$ we chose a value equal or greater than $\sigma^2_{\text{EKF to PF}}$ by trial and error. For the simulations the filter seems to be quite robust to changes in that value. If chosen too high, the filter might switch to late and diverge, which makes a global re-localization necessary.

6 Simulation Results

To explore and show the properties of the filter we have created five sets of simulated data and the corresponding maps. Each dataset is designed to show a certain property of the filter. The datasets were created with the simulation tool provided.

In the following sections we will present the important results of the simulations. However you will get a much better idea of the filter when you watch it run. Therefore we have prepared the code, that it is very easy to run the different simulations. See the file `README.txt` for instructions.

In all cases there is an additional white Gaussian sensor noise with standard deviation 0.01 rad for the bearing and 0.01 m for the distance.

You can find the complete settings for the simulation in the sub-folders of `./simulator/simfiles/`.

6.1 Passing Space without Measurements

In this simulation the robot starts at a position, where it has a couple of landmarks to localize. After a short time it travels straight ahead to another area with many landmarks and passes a longer space without any landmarks. As it reaches its goal it stops a short time to possibly localize and drives then to some spots with a high precision having many measurements all the time. After that it travels back to the position where it started again passing a distance without measurements and waiting a bit to relocalize.

One might imagine this scenario for a robot, which travels from its base station to some structure, which serves as landmarks, and performs some tasks there, which require high precision. In-between there are however no measurements.

We modeled the “true” wheel radius with 0.101 m 0.001 whereas the “modeled” value is 0.101 m for both cases, which suggests the robot that it constantly turns right if it sends controls to go straight. More realistic would be the setting that the robot turns right and thinks it goes straight but the implications for the estimator are the same and this way its easier to set up the simulation.

Note that this introduces a systematic error and thus violates the assumption of white process noise. In the labs we have seen that both the PF and the EKF can handle colored process noise, which is also important for the implementation in real systems.

We use parameters:

$\sigma^2_{\text{EKF to PF}}$	0.1
$\sigma^2_{\text{PF to EKF}}$	0.1
a	1
R_{PF}	$\text{diag}(10^{-4}, 10^{-4}, 3 \cdot 10^{-4})$
Q_{PF}	$\text{diag}(1, 1)$
λ_Ψ	0
M	1000
R_{EKF}	$\text{diag}(10^{-4}, 10^{-4}, 3 \cdot 10^{-4})$
Q_{EKF}	$\text{diag}(10^{-4}, 10^{-4})$
$\mu(t = 0)$	$(0, 0, 0)$
$\Sigma(t = 0)$	$10^{-10} \cdot I$
t_{pc}	0
$t_{\text{reloc\_window}}$	10.

You can find a plot with the simulation results in figure 1. The plots are similar to the ones produced in lab 1 and 2 (see [2], [3]).

We can see that the filter maintains a very good estimation of its position in the beginning. After having traveled a bit without measurements the filter switches to the PF until it reaches its destination where it switches back to EKF after some measurements. Moving around at the destination it remains a very good estimation of its pose. On the way back it again uses the PF to bridge the space without measurements.

If the EKF is ran alone on the same dataset it completely diverges, when it reaches the destination.

Under same conditions and without any visual output the simulation took 12.61 s while the running only the PF on the problem took 29.74 s. Thus the switching filter required less than half the processing power.

The error is also lower, which is mainly due to the high compensation noise and dominated by the phase without measurements. Therefore we don’t state the figures here.

The simulation can be run with the option `free_space` in `init.m`.

6.2 Include Constraints into the Particle Filter

This scenario is exactly the same as before apart from that way from the start point to the destination on the right of the map is only a small corridor between two walls. We could imagine the robot to have some short-range sensors such as IR sensors or bumpers to detect when it runs into a wall. Then only particles next to a wall could be maintained. Though in this simulation it doesn’t hit a wall. This concrete scenario is rather a bit constructed and should show the abilities of the PF and hence the switching filter to consider constraints.

The number of particles allowed to have been in wall when switching back to the EKF was chosen to be zero.

You can find a plot with the simulation results in figure 2.

We can see that the filter maintains a relatively good estimation when traveling without measurements, because all particles running into a wall get killed.

The simulation can be run with the option `free_space_with_walls` in `init.m`.

6.3 Global Localization

This scenario is pretty similar to the one in lab 2. We start with a global localization problem. Until the robot observes the rightmost landmark it has to track four hypotheses as the rest of the map is symmetrical. To show savings on computation power we let it run two rounds.

The wheels are again modeled as in section 6.1.

We use parameters:

$\sigma^2_{\text{EKF to PF}}$	0.1
$\sigma^2_{\text{PF to EKF}}$	0.1
a	1
R_{PF}	$\text{diag}(10^{-3}, 10^{-3}, 3 \cdot 10^{-3})$
Q_{PF}	$\text{diag}(1, 1)$
λ_Ψ	0
M	10000
R_{EKF}	$\text{diag}(10^{-4}, 10^{-4}, 3 \cdot 10^{-4})$
Q_{EKF}	$\text{diag}(10^{-4}, 10^{-4})$
δ_M	0.999
t_{pc}	0
$t_{\text{reloc\_window}}$	10.

M is chosen large to maintain the four hypotheses.

You can find a plot with the simulation results in figure 3. We see that the filter maintains all four hypotheses. As soon as it observes the landmark, which breaks the symmetry and has only one hypothesis left it switches to the EKF and keeps running the EKF for the rest of the simulation.

The EKF cannot handle a global localization problem. If its starting pose is just put somewhere it converges eventually in some cases in other is doesn’t.

The simulation for the switched filter runs 58.53 s which is significantly faster than the PF which runs 288.70 s.

The simulation can be run with the option `symmetric_asymmetric` in `init.m`.

6.4 Multiple Hypotheses Tracking

This scenario shows how different hypotheses can arise from a tracking problem due to ambiguous data association and how the filter handles this situation. In this case the measurements have only a short range of 1 m and the field of view is limited to 1.5 rad. We start with a tracking problem but have no measurements in the beginning. The huge uncertainty in the position is then reduced to at least two hypotheses. The situation is first unambiguous when the three landmarks at the right are observed.

The wheels are modeled as in section 6.1 except for the right wheel radius being 0.102 m in that case.

We use parameters:

$\sigma^2_{\text{EKF to PF}}$	0.01
$\sigma^2_{\text{PF to EKF}}$	0.1
a	5
R_{PF}	$\text{diag}(10^{-4}, 10^{-4}, 1 \cdot 10^{-2})$
Q_{PF}	$\text{diag}(0.1, 0.1)$
λ_Ψ	0
M	10000
R_{EKF}	$\text{diag}(10^{-4}, 10^{-4}, 3 \cdot 10^{-4})$
Q_{EKF}	$\text{diag}(10^{-4}, 10^{-4})$
δ_M	0.9999
$\mu(t=0)$	$(0,0,0)$
$\Sigma(t=0)$	$10^{-10} \cdot I$
t_{pc}	0
$t_{\text{reloc\_window}}$	10.

To properly get multiple hypotheses and account for the bigger wheel radius the third element of R_{PF} is chosen bigger in this setting.

You can find a plot with the simulation results in figure 4.

Shortly after the start the variance becomes big enough that the filter switches to the PF. It maintains the two hypotheses until it observes the one of the three landmarks at the right. As soon as it got rid of the other hypothesis it switches back to the EKF.

The vanilla EKF diverges in that case.

The simulation can be run with the option `tracking_symmetric` in `init.m`.

6.5 Kidnapped Robot Problem

This scenario should demonstrate the ability of the filter to do a global re-localization if lost. The landmarks are randomly distributed over the map area. We start with a global localization problem. The robot travels a while that the estimation can converge. Then it travels outside the area with landmarks and back into the area. Then the robot pose gets suddenly changed to (5,5,3.14). Then it travels again a while in the landmark populated area.

The wheels are again modeled as in section 6.1.

We use parameters:

$\sigma^2_{\text{EKF to PF}}$	0.08
$\sigma^2_{\text{PF to EKF}}$	0.08
a	1
R_{PF}	$\text{diag}(10^{-4}, 10^{-4}, 3 \cdot 10^{-4})$
Q_{PF}	$\text{diag}(1, 1)$
λ_Ψ	0
M	10000
R_{EKF}	$\text{diag}(10^{-4}, 10^{-4}, 3 \cdot 10^{-4})$
Q_{EKF}	$\text{diag}(10^{-4}, 10^{-4})$
δ_M	0.999
$\mu(t=0)$	$(0,0,0)$
$\Sigma(t=0)$	$10^{-10} \cdot I$
t_{pc}	0
$t_{\text{reloc\_window}}$	5.

You can find a plot with the simulation results in figure 5.

We see that the filter successfully localizes and switches to the EKF at time step 100. When it travels outside the area with landmarks it switches to the PF on the way back because the variance got too large. When the robot gets kidnapped it triggers a global localization after five time steps, which we chose as number of frames which have to fail the χ^2 test. It successfully re-localizes and switches back to the EKF.

When only the EKF is run on the problem it diverges when the robot gets kidnapped. It however doesn't diverge when traveling outside the area with the landmarks. So we could have chosen $\sigma^2_{\text{EKF to PF}}$ higher in that case. We didn't do that to demonstrate the two different mechanisms of switching to the PF.

The PF ran 212.64 s on this problem which way faster than the switched filter which ran 62.99 s.

The simulation can be run with the option `kidnapped` in `init.m`.

7 Conclusion

So we have seen that by switching between the PF and EKF on one estimation task depending on the belief distribution can be employed to have the versatility of the PF combined with the computational efficiency and the (theoretical) lack of discretization errors of the EKF. For the five simulations this concept worked very well. Especially the ability to relocate the EKF is one of the strengths of this concept.

Disadvantages are that two filters have to be implemented. As the filters are quite standard there might be ready-made code however as in our case. If not they are relatively simple to implement and share a good part of the code.

A more relevant disadvantage is that this concept requires more effort to be configured. One could possibly draw some directions from the steady state variance of the EKF at certain states. However we guess as some simulation or in field testing of an estimator is any necessary the proposed method to run the PF only will be the best choice in most cases. In case there is always the possibility to switch a bit "too late" to the EKF. Also some kind of auto tuning could be possible, where the filter switches relatively early to the EKF but the PF running for a while in parallel. If the estimations of the PF and the EKF diverge and the belief variance of the PF went down the EKF is initialized one more time. This is repeated until the estimations are consistent.

It might be interesting to compare this estimator to a PF, which adapts its noise settings and/or amount of particles according to its particle density (particles divided by belief variance). This could lead to similar properties.

Furthermore one could try to adapt this concept to the SLAM problem and switch between different estimators as for instance in [1], where an UKF is used instead of an EKF.

The results of this work are based purely on simulated data. Therefore the concept should be tested on real system.

References

[1] A. Dean, J. Langelaan, and S. Brennan. Improvements in terrain-based road vehicle localization by initializing an unscented kalman filter using particle filters. In *American Control Conference (ACC), 2010*, pages 700 –707, 30 2010-july 2 2010.

[2] J. Folkesson. Applied estimation(el2320) lab 1 ekf, 2011.

[3] J. Folkesson. Applied estimation(el2320) lab 2 instructions, 2011.

[4] S. V. U. Ha and J. W. Jeon. Combine kalman filter and particle filter to improve color tracking algorithm. In *Control, Automation and Systems, 2007. ICCAS '07. International Conference on*, pages 558 –561, oct. 2007.

[5] H. J. Lefebvre de Plinval-Salgues. Analysis of relative navigation architectures for formation flying spacecrafts. Master's thesis, Massachusetts Institute of Technology. Dept. of Aeronautics and Astronautics, 2006.

[6] L. Montesano and J. Gaspar. Fusing vision-based bearing measurements and motion to localize pairs of robots. In *In International Conference on Robotics and Automation*, 2005.

[7] L. Perea, J. How, L. Breger, and P. Elosegui. Nonlinearity in sensor fusion: Divergence issues in EKF, modified truncated SOF, and UKF. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Aug 2007 (AIAA-2007-6514).

[8] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

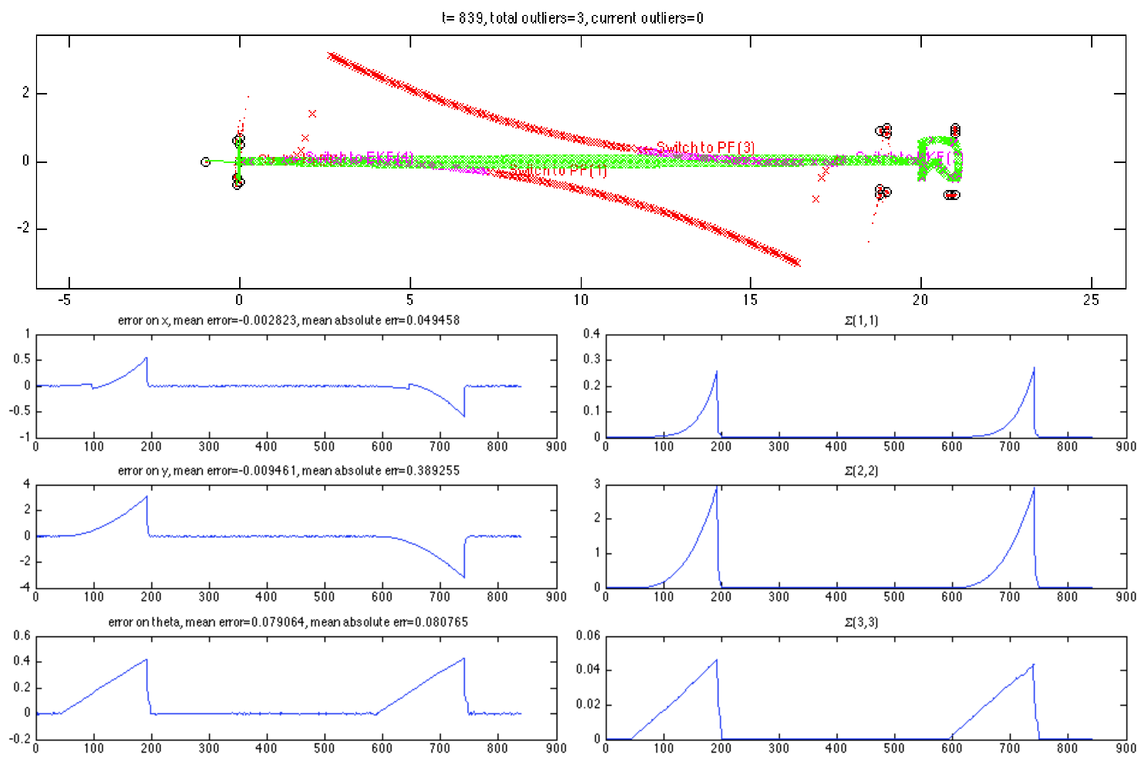

Figure 1: Simulation output for the scenario “Passing Space without Measurements”.

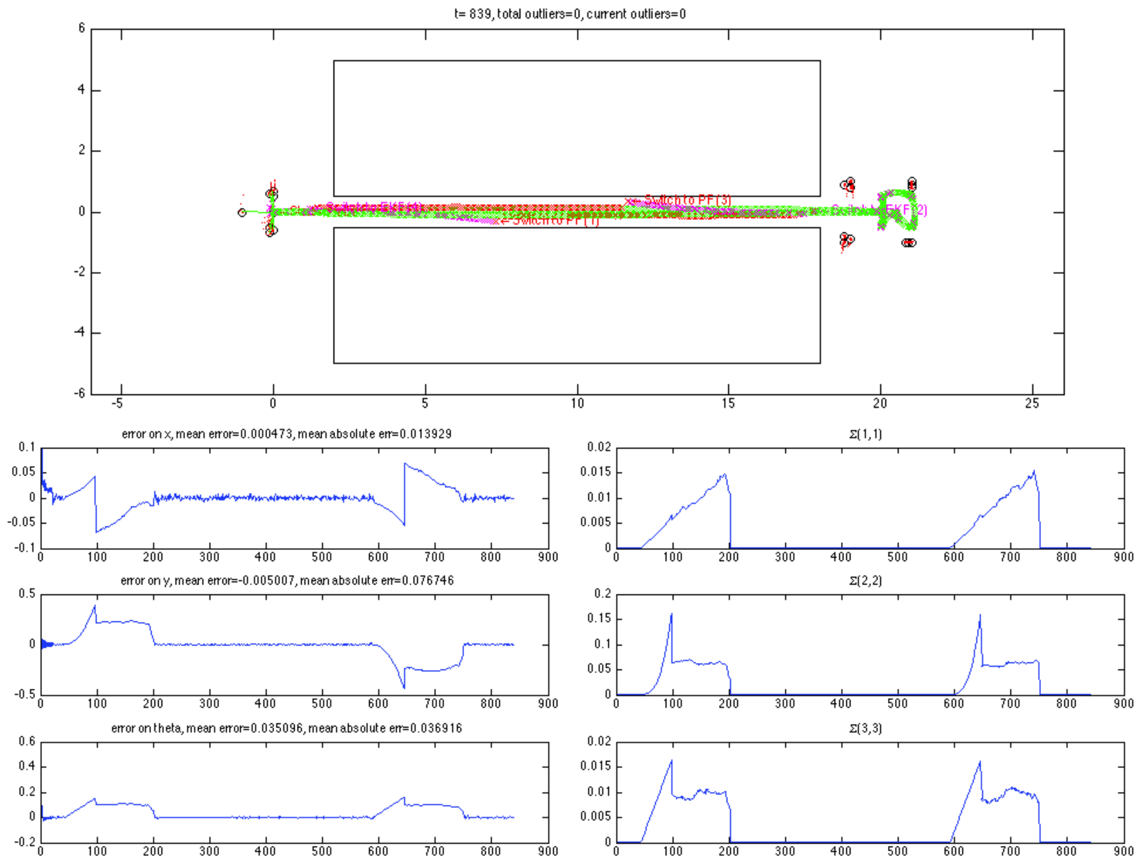

Figure 2: Simulation output for the scenario “Include Constraints into the Particle Filter”.

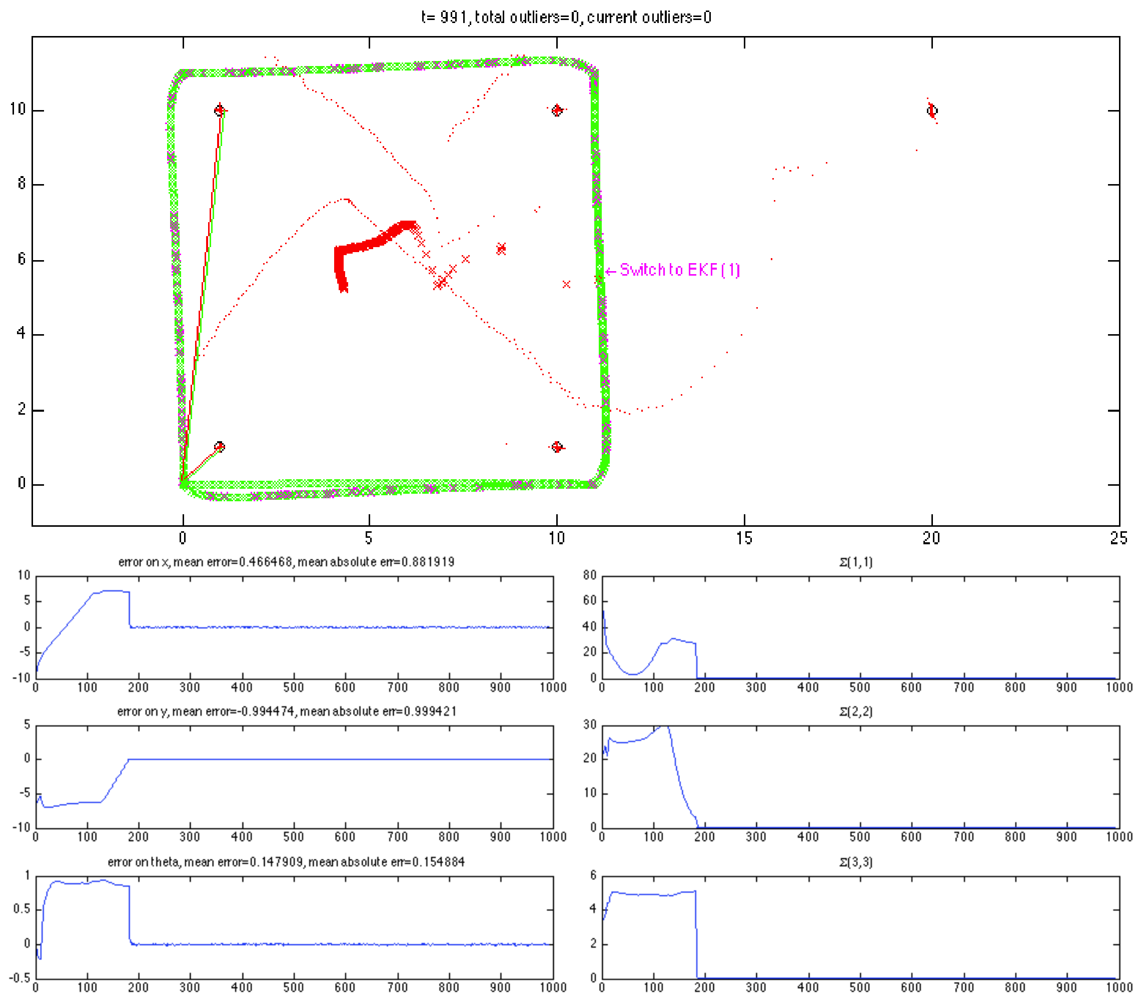

Figure 3: Simulation output for the scenario “Global Localization”.

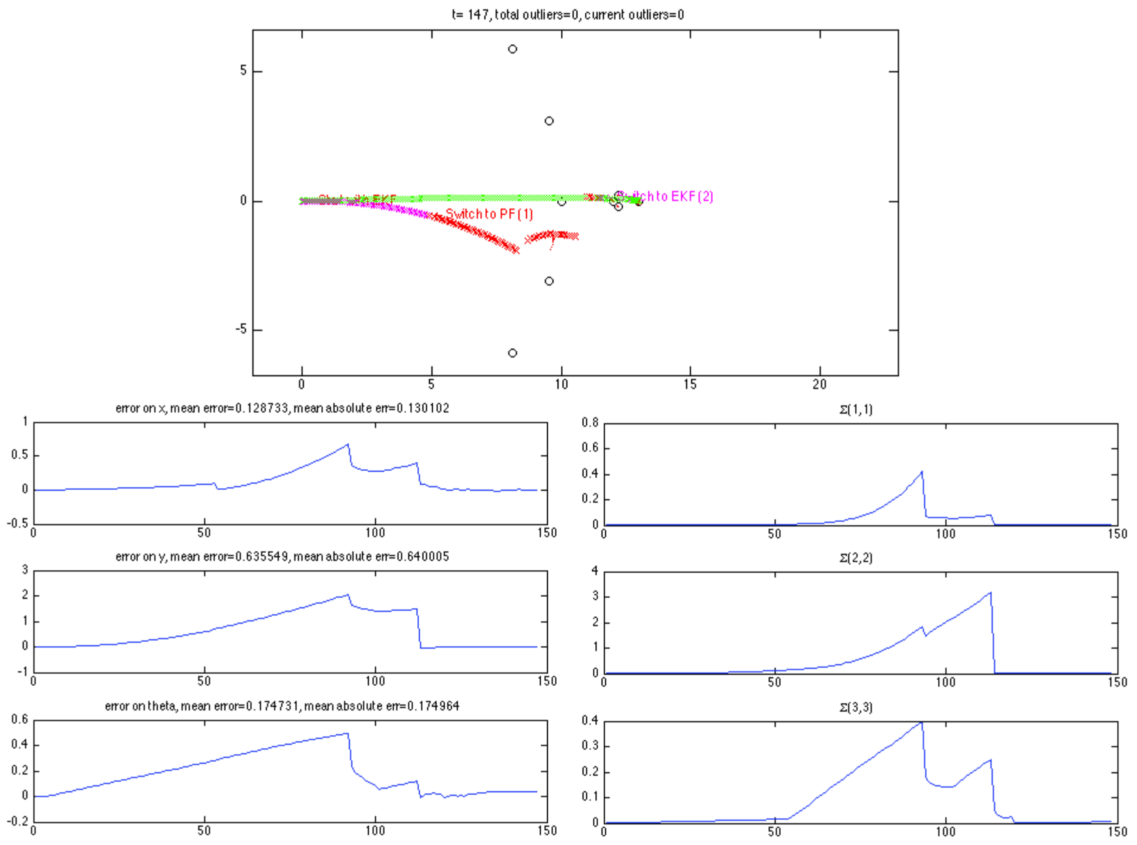

Figure 4: Simulation output for the scenario ”Multiple Hypotheses Tracking”.

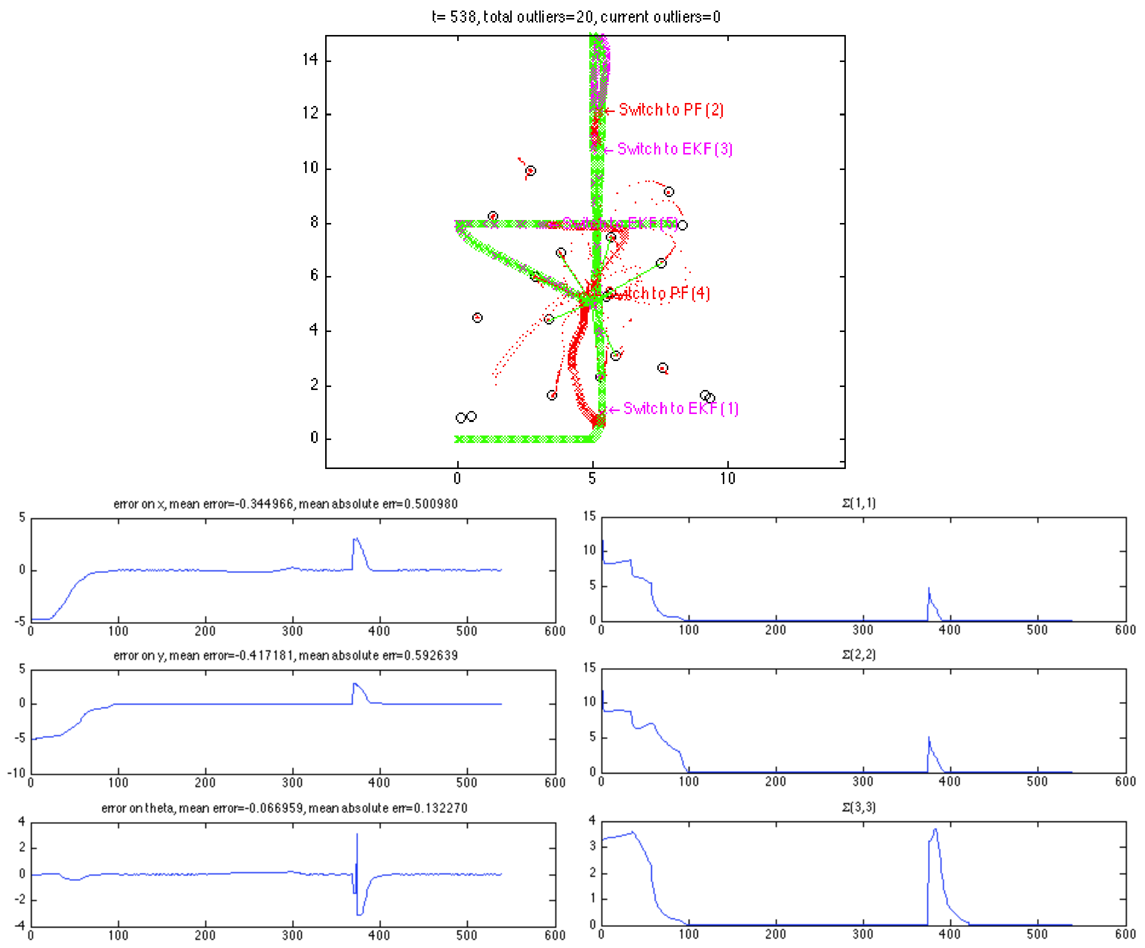

Figure 5: Simulation output for the scenario “Kidnapped Robot Problem”.