Real-Time Implementation of a Random Finite Set Particle Filter

Stephan Reuter, Klaus Dietmayer
Institute of Measurement, Control, and Microtechnology
University of Ulm, Germany
stephan.reuter@uni-ulm.de

Sebastian Handrich
Institute for Electronics, Signal Processing and Communications (IESK)
Otto-von-Guericke-University Magdeburg, Germany

Abstract: In scenarios characterized by a high object density, data association is a demanding task due to several ambiguities. Especially the assumption that all objects move independent from each other may lead to physically impossible predicted states, in which two objects are closer to each other than feasible. Thus, avoiding such impossible states may simplify the data association. Within the random finite set statistics it is possible to easily incorporate constraints concerning object states and to integrate them into a multi-target Bayes filter. A drawback of the random finite set statistics is its computational complexity, especially in the corrector step. In this contribution, a fast approximation for the calculation of the multi-target likelihood function is proposed. This approximation is used to implement a real-time random finite set particle filter on a graphical processing unit using real world sensor data.

1 Introduction

In many applications, object individual single-target trackers are used to track multiple objects. Thus, a data association algorithm is necessary in order to decide which measurement belongs to which single-target tracker. Especially in situations characterized by high clutter rates or high object densities, data association is often ambiguous. Thus, using simple algorithms like a nearest neighbor approach [BP99] may lead to poor tracking results since false associations are irreversible. Hence, algorithms like, e.g., Joint Integrated Probabilistic Data Association (JIPDA) [ME04] and Multi-Hypotheses Tracking (MHT) [BP99] have been developed, which avoid hard decisions by a probabilistic data association and propagation of all possible associations, respectively. Due to the propagation of all association hypotheses in MHT, the correct association is always represented by one of the hypotheses. A drawback is the huge amount of hypotheses that have to be handled. Especially in case of having a high number of targets and measurements, MHT gets computationally demanding.

In the multi-target Bayes filter proposed by Mahler [Mah07], a filter state represents the complete environment and not only one object. Thus, only a multi-target likelihood, which

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

is a measure of the affinity between the predicted environment and the received measurement set, has to be calculated. An association between measurements and tracks is not necessary in this approach. A drawback of the multi-target Bayes filter is the huge computational load which is necessary to calculate the multi-target likelihood during the corrector step.

In our scenario, two laser range finders are used to track several persons in an indoor scenario. The laser range finders are located in two corners of the room. For this scenario a Sequential Monte Carlo (SMC) implementation of the multi-target Bayes (MTB) filter was proposed in [RD11] which was not real-time capable. Further, an anti collision function, which keeps a minimum distance between any two persons, and a state dependent detection probability have been integrated. The MTB filter outperformed a CPHD filter [Mah07] especially in situations where some of the persons were occluded for a short time.

In this contribution, an approximation for the calculation of the multi-target likelihood function is presented. Additionally, an approximative calculation method for the state dependent survival probability is proposed. Based on these approximations, a parallel implementation of the MTB filter on a graphical processing unit (GPU) is introduced to speed up the MTB filter and achieve real-time performance.

The contribution is organized as follows: First, the multi-target Bayes filter is shortly reviewed. In Section 3 the approximative calculation method for the multi-target likelihood is introduced. Then, the calculation of the survival probability and implementation of the filter on a GPU are described in detail. Finally, in Section 6 the results of the approximative algorithm are compared to the results without using the approximation using real-world sensor data.

2 Multi-Target Bayes Filter

The MTB filter is an extension of the well known single-target Bayes filter. Instead of a single state $\mathbf{x}$ the multi-target Bayes filter propagates a random finite set $\mathbf{X}$ through time. An instantiation of random finite set consists of a random but finite number of state vectors belonging to different objects. Thus, it is random with respect to both, states and the number of states. In the following, we briefly summarize the multi-target Bayes filter. For more details about the multi-target Bayes filter we refer to [Mah07].

The predictor step of the MTB filter is given by

$$f_{k+1|k}(\mathbf{X}|\mathbf{Z}^{(k)}) = \int f_{k+1|k}(\mathbf{X}|\mathbf{X}') \cdot f_{k|k}(\mathbf{X}'|\mathbf{Z}^{(k)}) \delta\mathbf{X}', \tag{1}$$

where a set integral is used instead of the vector integral of the single-target filter [Mah07]. The multi-target Markov model of the predictor step has to represent not only the motion of the targets but also changes in the number of targets, as they may appear and disappear in the sensor's filed of view.

The corrector step of the multi-target Bayes filter is given by

$$f_{k+1|k+1}(\mathbf{X}|\mathbf{Z}^{(k+1)}) = \frac{f_{k+1}(\mathbf{Z}_{k+1}|\mathbf{X}) \cdot f_{k+1|k}(\mathbf{X}|\mathbf{Z}^{(k)})}{f_{k+1}(\mathbf{Z}_{k+1}|\mathbf{Z}^{(k)})} \tag{2}$$

with the Bayes normalization factor

$$f_{k+1}(\mathbf{Z}_{k+1}|\mathbf{Z}^{(k)}) = \int f_{k+1}(\mathbf{Z}_{k+1}|\mathbf{X}) \cdot f_{k+1|k}(\mathbf{X}|\mathbf{Z}^{(k)}) \delta\mathbf{X}. \tag{3}$$

The multi-target likelihood function $f_{k+1}(\mathbf{Z}_{k+1}|\mathbf{X})$ used in the corrector step is discussed in detail in section 3.

The multi-target Bayes filter can be implemented using Sequential Monte Carlo (SMC) methods [SW03, Mah07, RD11]. Here, a collection of multi-target particles

$$\mathbf{X}^1_{k|k}, \ldots, \mathbf{X}^\nu_{k|k} \tag{4}$$

and positive weights $w^1_{k|k}, \ldots, w^\nu_{k|k}$ with $\sum_{i=1}^{\nu} w^i_{k|k} = 1$ approximate the multi-target posterior distribution $f_{k|k}(\mathbf{X}|\mathbf{Z}^{(k)})$. Since each of the multi-target particles is an instantiation of a random finite set, each multi-target particle may consist of a different number of state vectors. As in the single-target case, the particles tend to degenerate. Thus, well-known resampling strategies like, e.g., importance sampling [RAG04] have to be used to avoid degeneracy.

3 Approximation of the Multi-Target Likelihood

In order to calculate the corrector step, the multi-target likelihood function $f_{k+1}(\mathbf{Z}_{k+1}|\mathbf{X})$ has to be evaluated. The multi-target likelihood function averages over all association hypotheses $\theta : \{1,\ldots,n\} \to \{0,1,\ldots,m\}$, where n is the number of states and m is the number of measurements. Since there is no a priori knowledge available on the correct association, all hypotheses are taken into account with an identical weighting factor. In case that only missed detections occur and no false alarms, the likelihood function is given by [Mah07]

$$f_{k+1}(Z|X) = f_{k+1}(\emptyset|X) \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_D(\mathbf{x}_i) \cdot f_{k+1}(\mathbf{z}_{\theta(i)}|\mathbf{x}_i)}{1 - p_D(\mathbf{x}_i)}, \tag{5}$$

where each association hypothesis is represented by one summand and $p_D(\mathbf{x}_i)$ is the detection probability. The factor $f_{k+1}(\emptyset|X)$ is given by

$$f_{k+1}(\emptyset|X) = \prod_{i=1}^{n} \left(1 - p_D\left(\mathbf{x}_i\right)\right), \tag{6}$$

where $\emptyset$ denotes the empty set.

In a SMC implementation, the multi-target likelihood has to be calculated for all multi-

Multiobjekt Partikelfilter - Innovation

Track - Messung

$\theta : \{1, \ldots, n\} \rightarrow \{0, 1, \ldots, m\}$

- Likelihood-Funktion kann als Baum implementiert werden

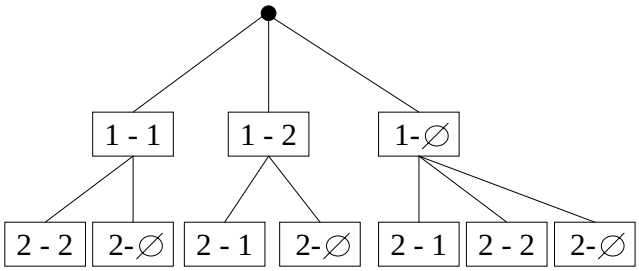


Figure 1: Tree structure to evaluate the multi-target likelihood function: first value of each node represents the index of the particle, the second value represents the number of the measurement or the missed detection ($\emptyset$).

$$f_{k+1}(Z|X) = f_{k+1}(\emptyset|X) \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_D(\mathbf{x}_i) \cdot f_{k+1}(\mathbf{z}_{\theta(i)}|\mathbf{x}_i)}{1 - p_D(\mathbf{x}_i)}$$

Since the complexity of the association tree for each multi-target particle increases approximately exponentially in the number of measurements, an approximation of the calculation is necessary to achieve real-time performance. In Fig. 1 the possible associations for the second particle in the set depend explicitly on the association for the first particle due to the assumption that one measurement is generated by at most one target. Thus, each path of the tree has to be calculated separately.

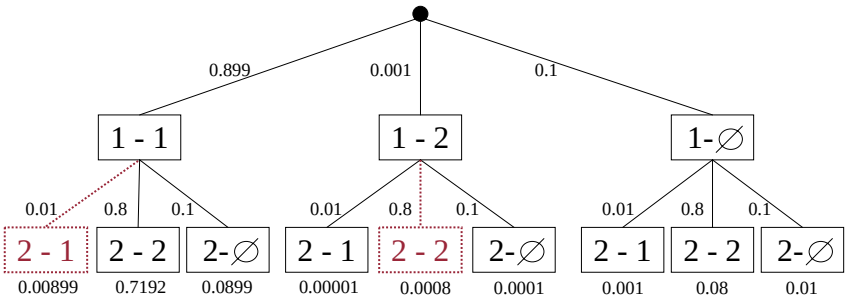
To simplify calculations, we propose to neglect the assumption that a measurement is generated by at most one target during the construction of the tree. This leads to the approximate tree shown in Fig. 2. Here, the possible associations for the second particle in the set do not depend on the associations for the first particle any more. In this example, the exact value of multi-target likelihood is approximately 0.901 while the result of the approximation is 0.91. Thus, there is only a small deviation of roughly one percent.

Using this approximation, the calculation of the multi-target likelihood simplifies to

$$f_{k+1}(Z|X) = f_{k+1}(\emptyset|X) \prod_{i=1}^{n} \left(1 + \sum_{j=1}^{m} \frac{p_D(\mathbf{x}_i) \cdot f_{k+1}(\mathbf{z}_j|\mathbf{x}_i)}{1 - p_D(\mathbf{x}_i)} \right). \tag{7}$$

The sum in this equation represents all possible associations for particle n of the set. The product multiplies the association values of all particles. The complexity of equation (7) increases only linear in the number of tracks as well as in the number of measurements while the complexity of (5) increases approximately exponentially. Further, equation (7) can be implemented using two for-loops instead of a recursive function.

Approximation des Baumes



- Exakter Wert: 0.9010
- Approximativer Wert: 0.91
- Relativer Fehler: ungefähr 1%
- Vorteil: Berechnungen unabhängig von vorherigem Pfad

Figure 2: Approximation of the multi-target likelihood tree: Nodes with dashed lines are added to the graph to remove dependency on associations of previous stages. Each link has a likelihood value and the multi-target likelihood is calculated by summing up the values at the bottom. This example uses a constant detection probability of $p_D = 0.9$.

In our scenario, the size of a person is approximately as large as the 3 σ error ellipse of the measurements. Since the multi-target particles represent the whole environment, it is possible to integrate an anti-collision function [RD11] into the prediction step, which keeps a minimum distance between any two particles in the set, to avoid physically impossible predictions. Fig. 3 shows one measurement located exactly in the middle between two objects. Due to the necessary minimum distance between the objects, the spatial likelihood

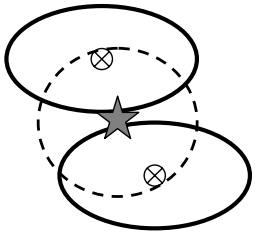


Figure 3: Two objects (ellipses) and their centers (crosses) and one measurement (star) with according 3 σ bound.

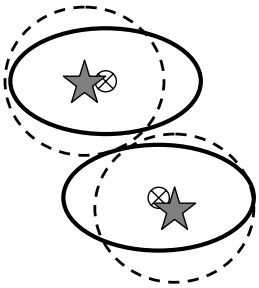


Figure 4: Two objects (ellipses) and their centers (crosses) and two measurements (stars) with according 3 σ bound.

of the measurement is very small for both of the objects. Fig. 4 shows two measurements located close to the centers of the two objects. In this case, the spatial likelihoods for the correct association are very high. Further, only one of the objects is in the 3 σ bound of the measurements. Thus, the expected error in the calculation of the multi-target likelihood is considered to be very small in our application.

4 Detection and Survival Probability

Due to the small distance between the pedestrians and the sensors, each of the pedestrians causes a large area which is occluded for the sensor. Thus, it is necessary to use a state dependent detection probability. The adapted method of the occupancy grid mapping approach [TBF05] introduced in [RD11] is used to calculate these detection probabilities.

Further, the calculated occupancy grid allows to integrate state dependent survival probabilities. In observable areas, a survival probability of $p_S = 0.995$ is used. Obviously, a pedestrian can not be located at grid cells where the occupancy grid is occupied by a static obstacle. Thus, a survival probability of $p_S = 0.1$ is assigned to a particle located in one of these cells. The occupancy grid also provides us the information, which grid cells have never been observed by the sensor up to the current time step. In order to avoid objects spinning around in the unobservable area for a long time, the survival probability for these grid cells is set to $p_S = 0.8$. Thus, it is possible that an object crosses a small unobservable area without the need of re-initialization. The survival probabilities for the three states have been determined experimentally by comparing the results of the tracking algorithm for several different values.

5 GPU Implementation of the Multi-Target Bayes Filter

Particle filters are very suitable for parallel processing, since the prediction and corrector step of each particle are independent of all the other particles. Thus, a graphical processing unit (GPU) is used to compute the computationally intensive parts of the particle filter. The filter is implemented using CUDA [Nvi11]. The Nvidia CURAND library [Nvi11] is used to generate the necessary random numbers on the GPU.

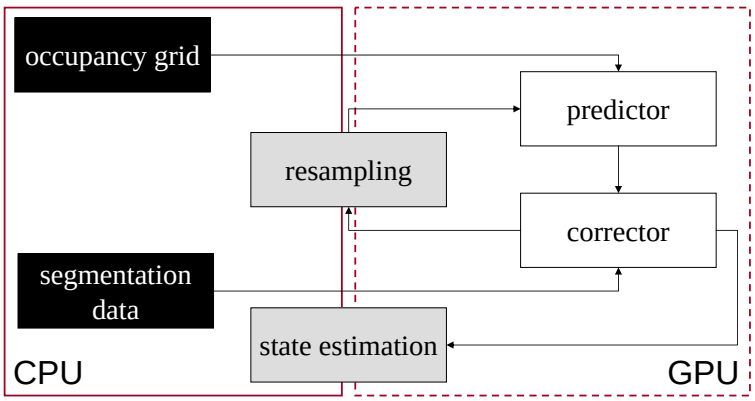


Figure 5: Distribution of functions between CPU and GPU. Blocks with black background are calculated on the CPU, white ones on GPU. Computation of gray colored boxes is partly on CPU and on GPU.

Figure 5 shows an overview of the implemented system. The CPU calculates the occupancy grid and the associated detection probability map and uploads them to the GPU. Further, the CPU generates segments based on the fuzzy segmentation algorithm introduced in [RD09] using the raw measurements of the laser range finders and transmits them to the GPU. The GPU is used to calculate the predictor and the corrector step of the multi-target Bayes particle filter.

The predictor step consists of three parts: motion, persistence and appearance. The multi-target particles are predicted to the next time step using a standard Markov model which assumes independent motion of all the objects in the multi-target particle. Then, the persisting state vectors of a multi-target particle are determined by drawing a sample of a multi-target multi-Bernoulli distribution [Mah07]. The anti collision function introduced in [RD11] adapts the weights of the persisting multi-target particles based on the minimum distance between any two of the objects in the multi-target particle. A measurement driven appearance model creates birth candidates for each multi-target particle based on the previous measurement set and the multi-target particle. Each measurement, which is outside the 3 σ range of each state vector in a multi-target particle is considered as a birth candidate [RD11]. A birth candidate is added with a birth probability of $p_B = 0.001$ to the multi-target particle. Finally, the predicted multi-target particle is given by the union of the persisting and the appearing set:

$$\mathbf{X}^i_{k+1|k} \triangleq \mathbf{X}^{i,\text{persist}}_{k+1|k} \cup \mathbf{X}^{i,\text{appear}}_{k+1|k}. \tag{8}$$

Due to the use of an appearance model, it is possible to initialize the filter using a zero-target prior, i.e. there are no objects in the scene.

The corrector step calculates the approximative multi-target likelihood function introduced in Section 3 for each of the multi-target particles. In order to avoid numerical problems for large numbers of measurements, equation (7) is calculated in the logarithmic domain.

In the state extraction step, GPU and CPU share the calculations. The number of targets $\hat{n}$ is estimated by calculating the weighted mean

$$\hat{n}_{k+1|k+1} \cong \frac{1}{\nu} \cdot \sum_{i=1}^{\nu} w^i_{k+1|k+1} \cdot |\mathbf{X}^i_{k+1|k+1}| \tag{9}$$

on the CPU, where $|\mathbf{X}^i_{k+1|k+1}|$ is the number of states of the multi-target particle. Since two multi-target particles with the same number of objects $\tilde{n}$ do not have to have the same order of the objects or the same $\tilde{n}$ objects, a state estimation using a weighted mean can not be applied directly. Thus, a grid map is implemented on GPU for a PHD-based state estimation [Mah07]. For each of the state vectors in the multi-target particles, the according grid coordinates are calculated and the value of the grid cell is increased by the weight of the multi-target particle. Then, the $\hat{n}_{k+1|k+1}$ highest peaks of the grid deliver the estimated states.

The resampling step is again shared between CPU and GPU. First, the weight vector is transmitted from GPU to CPU. The CPU uses the weight vector to calculate a vector which holds the indexes of the samples to draw. Then, the GPU uses the indexes to perform the resampling step.

ISBN 978-3-88579-286-4

6 Results

The multi-target Bayes filter is applied to real-world sensor data of two laser range finders. In the scene, up to seven pedestrians with up to three pedestrians per square meter occur. Due to the large number of measurements, the measurements are clustered. First, the measurements of each sensor are clustered using a fuzzy segmentation algorithm [RD09]. The fuzzy segments of the sensors are combined using cross-validation. Since the torso of a pedestrian has an approximately elliptical shape, the least squares approach introduced in [FF99] is used to fit ellipses into the clusters. Fig. 6 shows the results of the applied clustering algorithm. We observe, that each of the five persons generates at least one

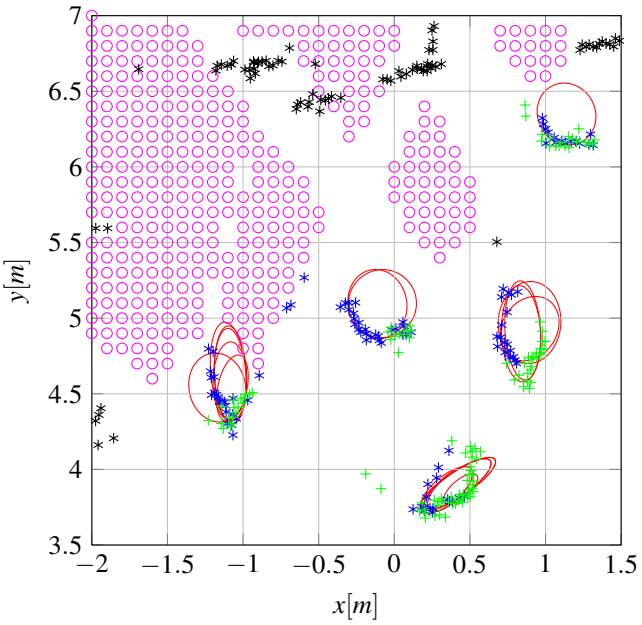


Figure 6: Segmentation Results: Estimated locations and sizes of the persons are illustrated by the red ellipses. Blue stars are measurements of sensor 1, green stars of sensor 2. The black stars are measurements of non-moving objects. The area which is occluded for both sensors is marked by magenta circles.

segmentation result. Further, the range measurements of the sensors are separated into stationary and non-stationary measurements using the occupancy grid and the occluded area is marked by magenta circles. Currently, the implemented SMC-MTB filter does not use the size of the objects. Thus, the centers of the ellipses are used as measurements.

Fig. 7 shows the estimated states of the approximative (cyan) and the exact (blue) multi-target likelihood calculation of a MTB filter with 10000 multi-target particles. The EM-clustering algorithm [DLR77] is used in this figure to estimate the states of the objects as well as the according state uncertainties. Although the three pedestrians at the top are very

close to each other, the results of the approximation are almost identical to the ones of the exact calculation. The usage of the anti collision function in the prediction step is in this situation an important factor to achieve the accurate estimation results.

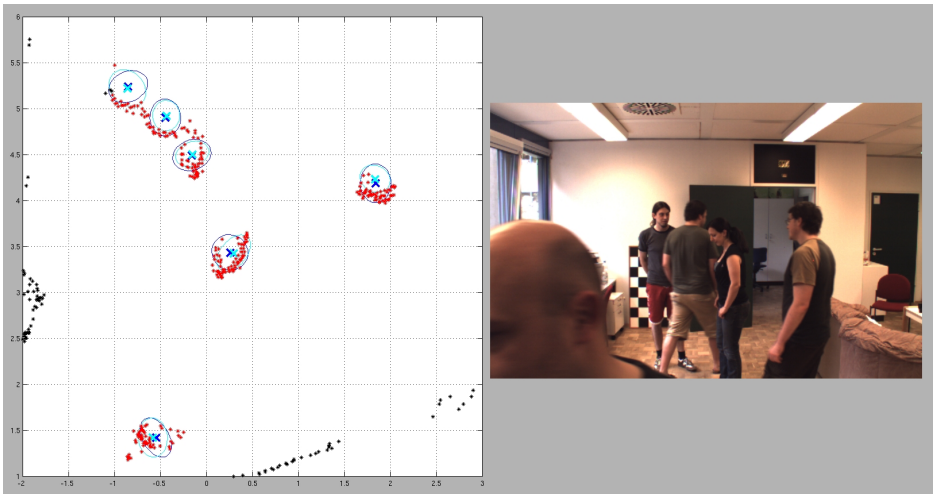


Figure 7: Comparison of approximative and exact multi-target likelihood calculation: Estimated object states of approximative (cyan) and exact (blue) calculation marked by crosses. Ellipses illustrate the uncertainties. Red stars are the raw range data of the laser range finder.

Fig. 8 shows the mean value of the estimated number of objects over 18 Monte Carlo runs. The mean value is compared to the total number of pedestrians in the scene and the number of pedestrians which can be observed by the sensors. The filter needs approximately 150 cycles (12 seconds) to initialize to the correct number of objects due to some occlusions during this time interval and a small birth probability. After the initialization, the mean value for the number of pedestrians is very close to the number of pedestrians in the scene, if the pedestrians are only occluded for a short time (up to approximately 0.7 seconds). During long-time occlusions (e.g. for approx. $380 < k < 460$), the estimated number of objects slightly decreases. Since some of the multi-target particles still represent the proposition that there are 6 objects in the scene, the estimated number of objects rapidly increases to the true number of objects after the occlusion. At $k \approx 1180$ and $k \approx 1250$ we observe the effect, that the mean value of the estimated number of objects increases after occlusions. This is caused by the fact, that some of the particles of the occluded person are located outside of the measurement uncertainty in an occluded area. In order to mitigate this effect, a more sophisticated birth model has to be developed. At $k \approx 730$ and $k \approx 1330$, the estimated number of objects drops rapidly for a short time due to several false negative measurements in a row for one of the objects.

Fig. 9 compares the results of the GPU implementation to the results of the CPU implementation. While the GPU uses the approximative calculation of the multi-target likelihood, the CPU uses the exact calculation. For $k < 235$, the results of both implementations

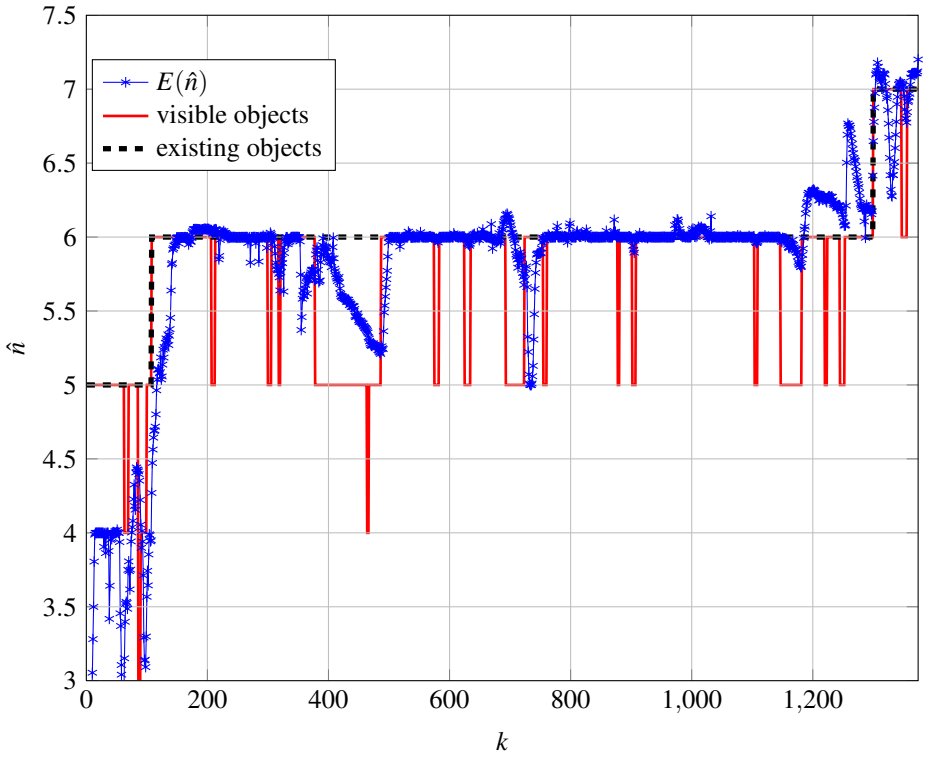


Figure 8: Estimated number of pedestrians: Mean value of the estimated number of objects over 18 Monte Carlo runs (blue stars). Further, the total number of pedestrians in the scene (dashed line) and the number of visible pedestrians (solid line) are shown.

are almost identical. During the occlusion of one of the pedestrians for $235 \leq k < 270$, the estimated number of objects of the GPU implementation is closer to the ground truth than the one of the CPU implementation. When the pedestrian re-appears in the field of view at $k = 270$ and $k = 341$, the CPU implementation delivers better results than the GPU version. This is caused by the fact, that the predicted particles of the occluded pedestrian are to far away from the received measurement at the time of re-appearing. Thus, the measurements lead to the appearance of an additional object and consequently the estimated number of objects becomes higher than the actual number of objects. Since the minimum distance between any two of the pedestrians is at least about 5σ at the time of re-appearance, it is unlikely, that the difference between GPU and CPU implementation is mainly caused by the approximation of the multi-target likelihood. Thus, other factors like the usage of different random number generators on GPU and CPU might be responsible for this effect, too.

On a Core2-Quad CPU with 2.83 GHz, one cycle of the MTB filter (prediction, correction, state estimation, resampling) with 10000 multi-target particles takes around 500 milliseconds for 10 measurements and 6 tracks. Due to the measurement rate of 12.5 Hz, it is not

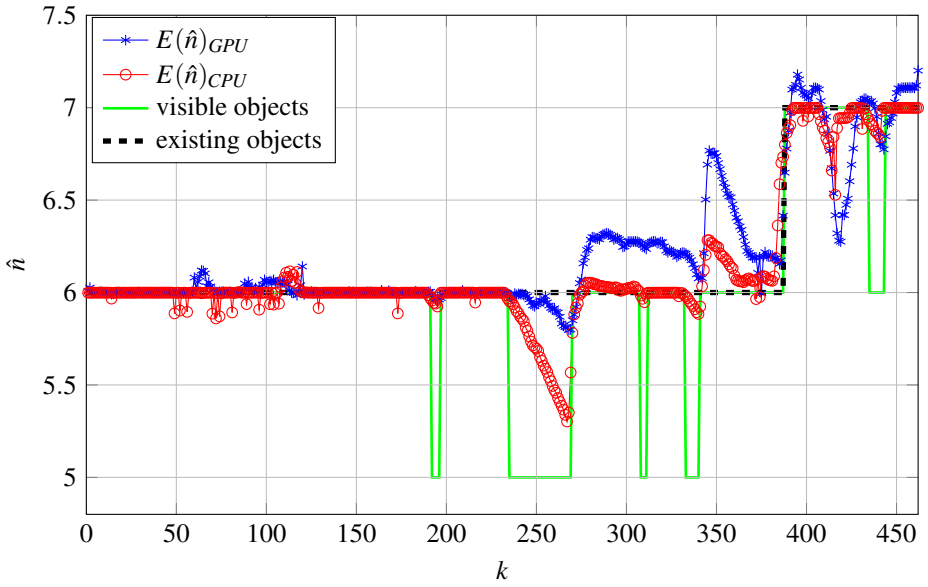


Figure 9: Estimated number of pedestrians: Mean value of the estimated number of objects over 18 Monte Carlo runs for GPU (blue stars) and CPU (red circles) algorithm. GPU version calculates the approximative multi-target likelihood, CPU version calculates the exact multi-target likelihood. Further, the total number of pedestrians in the scene (dashed line) and the number of visible pedestrians (solid line) are shown.

possible to use this implementation in a real-time application. On a GPU, one cycle of the MTB filter takes only approximately 4 milliseconds for the same calculation, which is approximately 100 times faster. Due to the linear complexity of the approximative multi-target likelihood, an application of the algorithm to a larger number of measurements and tracks is possible without losing the real-time capability.

7 Conclusion

In this contribution a real-time implementation of a random finite set particle filter on a graphical processing unit has been presented and applied to real world sensor data. To achieve real-time performance, an approximative calculation of the multi-target likelihood has been introduced which dramatically reduces the complexity from exponential to linear. Due to the usage of an anti collision function in the predictor step, the results of the exact and the approximative multi-target likelihood calculation are almost identical.

In future, we plan to integrate more interactions between the pedestrians and the environment by applying more sophisticated models. Further, the performance of the random finite set particle filter will be evaluated regarding the state accuracy. In order to improve the performance of the filter concerning the estimated number of objects, an enhanced

modeling of the birth and survival probabilities has to be developed.

ACKNOWLEDGMENTS

This work is done within the Transregional Collaborative Research Center SFB/TRR 62 ”Companion-Technology for Cognitive Technical Systems” funded by the German Research Foundation (DFG).

References

[BP99] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[FF99] M. Fitzgibbon, A. W.and Pilu and R. B. Fisher. Direct Least-Squares Fitting of Ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999.

[Mah07] Ronald P.S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House Inc., Norwood, 2007.

[ME04] D. Musicki and R. Evans. Joint Integrated Probabilistic Data Association: JIPDA. *IEEE Transactions on Aerospace and Electronic Systems*, 40(3):1093 – 1099, july 2004.

[MMD10] Michael Munz, Mirko Mählisch, and Klaus Dietmayer. Generic Centralized Multi Sensor Data Fusion Based on Probabilistic Sensor and Environment Models for Driver Assistance Systems. *IEEE Intelligent Transportation Systems Magazine*, 2(1):6–17, 2010.

[Nvi11] Nvidia. Nvidia CUDA Toolkit 4.0 - available online: http://developer.nvidia.com/cuda-toolkit-40, 2011.

[RAG04] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Inc., 2004.

[RD09] Stephan Reuter and Klaus Dietmayer. Fuzzy Estimation and Segmentation for Laser Range Scans. In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, USA, July 2009.

[RD11] S. Reuter and K. Dietmayer. Pedestrian Tracking Using Random Finite Sets. In *Proceedings of the 14th International Conference on Information Fusion*, 2011. accepted for publication.

[SW03] Hedvig Sidenbladh and Sven-Lennart Wirkander. Tracking Random Sets of Vehicles in Terrain. In *Conference on Computer Vision and Pattern Recognition Workshop*, 2003.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.