2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

MuSe: Multi-Sensor Integration Strategies Applied to Sequential Monte Carlo Methods

Richard Hanten, Cornelia Schulz, Adrian Zwiener and Andreas Zell

***Abstract*— Recursive state estimation is often used to estimate a probability density function of a specific state, e.g. a robot's pose, over time. Compared to Kalman filters, Sequential Monte Carlo (SMC) methods are less constrained in regard to state propagation and update model definition, which makes it easier to implement any suitable problem. In this work, we present a generic Sequential Monte Carlo framework, which uses abstract formulations for importance weighting, propagation and resampling and provides an independent core algorithm that is usable for any problem instantiation, such that diverse SMC problems can be implemented easily and quickly, since the basic algorithms are already provided. Current applications include 2D localization, 2D tracking in a SLAM system and contact point localization on a manipulator surface. Further, we introduce concepts to deal with data input synchronization and fair execution of different weighting models, which makes it possible to incorporate data from as many update sources, e.g. sensors, as desired. As a typical application scenario, we provide a plugin-based and hence easily extensible instantiation for 2D localization and demonstrate the capabilities of our framework and methods based on a well-known dataset.**

I. Introduction

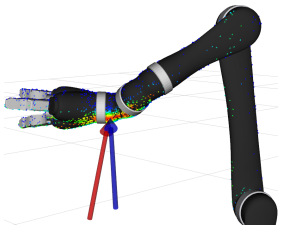
Many robotic tasks require the estimation of a specific state, which can be deduced from observations that depend on it. The most common problem is the localization of a mobile robot. To improve reliability and robustness, many localization approaches are implemented in a probabilistic manner, allowing to model noisy sensor measurements and different influences from the robot's surrounding. These methods usually involve either Kalman filters or Sequential Monte Carlo methods, commonly called particle filters.

Generally, it is reasonable to integrate as much sensor data as is available into the state estimation process to derive a consistent and stable estimate. While Kalman filters are most popular to fuse multiple sensors into different state spaces [1], [2], more and more approaches [3], [4], [5] use Monte Carlo Localization (MCL) in a multi-sensor scheme, because it can be implemented and extended easily.

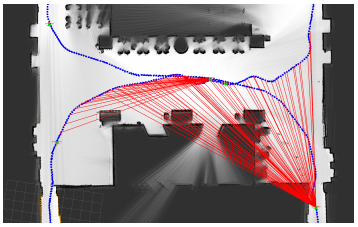
Further, the SMC computation pipeline can directly be transferred to solve any other recursive state estimation problem, for example contact point localization for manipulators or 2D pose tracking, as depicted in Fig. 1.

Since the SMC core and the basic components involved stay the same, we abstracted and separated them from the problem instantiation, which resulted in a powerful framework that can be used to rapidly implement SMC problems and is not restricted to robotic problems.

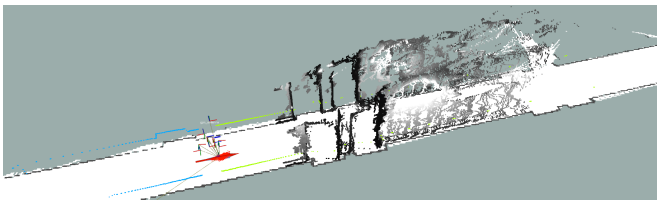
R. Hanten, C. Schulz, A. Zwiener and A. Zell are with the Cognitive Systems Group at the Computer Science Department, University of Tuebingen, Sand 1, 72076 Tuebingen, Germany.
Contact: richard.hanten@uni-tuebingen.de



(a) Contact Point Localization with proprioceptive sensors on a Kinova Jaco2. Shown are the particles (colored by likelihood), real (red) and estimated (blue) contact.



(b) A SLAM system using the SMC framework for pose tracking with dynamically updated maps. Part of a grid map and loop closure constraints (red) between poses (blue) and map origins are shown.



(c) 2D Monte Carlo Localization with two laser scanners and a stereo camera.

Fig. 1. Examples of application scenarios we applied our SMC framework to: (a) contact point localization on a manipulator surface, (b) 2D pose tracking in a SLAM context and (c) 2D localization of mobile robots.

The contributions of our work are the following:

- We present abstraction strategies that allow us to separate the Sequential Monte Carlo processing pipeline from the incorporated functions which depend on the specific problem space.
- To correctly integrate different kinds of sensor data into the SMC method, we also propose data synchronization and function execution scheduling principles. This way, the SMC method becomes multi-sensor capable.
- We integrated these concepts into an abstract SMC framework that can be used to implement different SMC methods easily. The framework is open source and can be found on GitHub[1].
- To evaluate the proposed concepts, we implemented a plugin-based and therefore easily extensible 2D MCL instantiation of this framework[2] and measured the impact of these concepts on the amount of sensor data that can be integrated and the resulting localization accuracy.
- Further examples where we already used our framework demonstrate its general applicability. For now, these are contact point localization on a mesh surface of a manipulator, 2D pose tracking in a SLAM context and 2D localization, as illustrated in Fig. 1.

[1] https://github.com/cogsys-tuebingen/muse_smc
[2] https://github.com/cogsys-tuebingen/muse_mcl_2d

II. Related Work

Since the introduction of Monte Carlo Localization by Fox et al. [6], there have been several works trying to improve the pose estimation process by employing novel models for particle set initialization, particle propagation and weighting based on diverse sensors or using additional feature- or landmark-based techniques.

Ito et al. [3] use the WiFi signal strength to limit the initial particle scattering across the whole map. Similarly, Miyagusuku et al [4] insert new particles based on the WiFi signal strength in case of localization failure, and they also combine a novel WiFi sensor model with a laser model by multiplying the particle weights. Perea et al. [5] did the same with GPS instead of WiFi and adapted the propagation step by replacing the least accurate particles by Gaussian distributed ones centered at the GPS position estimate.

State propagation does not necessarily depend on the availability of wheel odometry, but can be done using visual odometry as well [3]. Winterhalter et al. [7] use the native 6DoF visual pose estimation system integrated into the Google Tango platform, and also provide a sensor model that estimates surface normals from RGB-D data and checks them against a vector-based 2D floor plan. Similar, Biswas and Veloso [8] additionally apply plane fitting for input data filtering. Alternatively, floor plan localization was also already directly solved using vector maps and laser data [9].

More specific methods are based on landmark extraction. Elinas and Little generate 3D SIFT feature points from stereo matching and use a sensor model based on a 3D landmark map [10]. In a RoboCup soccer game, Röfer et al. [11] extracted landmarks like goals and edges between soccer field lines from camera images.

Many of these works are based on the 2D MCL implementation *AMCL* [12], which is part of the *Robot Operating System (ROS)* software landscape. Therefore, our 2D MCL framework instantiation is also roughly based upon AMCL and provides the same functionality, amongst others, but provides multi-sensor support by applying data synchronization methods similar to the work of Lynen et al. [1].

Furthermore, most of the approaches described above can be realized with our 2D MCL core by defining the specific prediction and weighting models as well as the map types in form of easily exchangeable plugins. This way, the development process of new MCL components like sensor models could be sped up massively.

Another abstract SMC framework is the *Sequential Monte Carlo Template Class (SMCTC)* [13], which has been applied to RGB-D based MCL [14]. In contrast to our framework, SMCTC only provides few abstract headers and the user needs to implement all desired functionality on his own, including the SMC core. Since our basic SMC framework does not depend on ROS, it can be used for non-robotic applications as well, and already provides implementations of the basic components like different resampling schemes.

The achieved level of abstraction made it possible to implement a particle filter method for contact point localization directly on a manipulator surface [15].

III. Concepts

In the following, we briefly summarize the main theoretical principles and essential equations needed for *Sequential Monte Carlo (SMC) Methods*. Then, we present the technique of abstraction we used to separate the SMC core algorithm from its instantiation and the applied functions. Finally, we introduce the synchronization and scheduling principles we needed to correctly incorporate multiple sensor inputs.

A. Sequential Monte Carlo Methods

One of the most popular SMC methods is *Monte Carlo Localization (MCL)*, an approach for robot pose estimation in known environments. Based on a map m, controls $u_{1:t}$ and measurements $z_{1:t}$, the filter provides the belief state

$$bel(x_t) = p\left(x_t \mid u_{1:t}, z_{1:t}, m\right) \propto X_t \tag{1}$$

as a set of weighted particles $X_t = \{x_t^1, x_t^2, \ldots\}$, from which each particle is a possible robot pose at time t.

To transfer the belief state of a discrete time stamp to the next one, MCL consists of two steps: a motion update and a sensor update. After some time, an additional resampling step is applied to improve the overall particle set likelihood.

1) Prediction: The particle set for the next time stamp is predicted by applying a motion model based on the current control u_t to the previous particle distribution:

$$x_t^i \sim p(x_t \mid x_{t-1}^i, u_t). \tag{2}$$

2) Update: Each particle is then weighted based on the likelihood that the current sensor readings z_t fit the map when using this particle as pose estimate:

$$w_t^i = p(z_t \mid x_t^i, m). \tag{3}$$

3) Resampling: To reduce the number of unlikely particles, a resampling step is used to replace the particle set by a more likely one. Therefore, new particles are drawn randomly from the previous ones, such that particles with high weights w_t^i are more likely to be introduced into the new set.

In order to deploy different SMC methods, merely the state space description of X as well as prediction and update functions need to be exchanged. The SMC filter pipeline stays the same.

B. Abstraction

To reach a level of abstraction which allows to employ a particle filter independently from data and state space representations as well as associated models, we use a technique commonly used in functional programming called *currying*.

For the propagation of the filter's belief state, a motion model $P(X, u)$ has to be applied. Binding the control input u_i at time stamp i to the function can be formally described as shown in the following equation:

$$P_{u_i}(X) = \lambda_P \left(P(X, u), u_i\right) \mid X \to P(X, u_i) \tag{4}$$

As a result, we obtain a function P_{u_i} which only takes a particle set X as an input. Analogously, the same can

be done for an update model $U(X, z, m)$, which applies an importance weight update:

$$U_{z_j,m_j}(X) = \lambda_U \left(U(X, z, m), z_j, m_j\right) \mid X \to U(X, z_j, m_j) \tag{5}$$

Binding a measurement z_j at time stamp j with the associated state space representation m_j, in case of MCL a map, to the model yields a function $U_{z_j,m_j}(X)$, which also solely takes the particle set as an input argument.

Since all functions only depend on the particle set, the Bayesian core algorithm can be completely separated from the propagation method and the sensor models. This concept is visualized in Fig. 2.

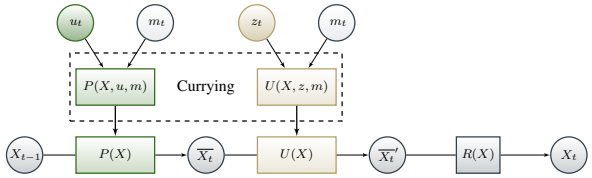


Fig. 2. Simplified illustration of the state estimate progress. Currying is applied to the prediction model P, binding control data u_t and potentially state space information m_t. In general, we assume several applications of the prediction model, with sequential control data. In addition, currying is applied to the update model U, binding sensor data z_t and information m_t to the model. The belief state of X_t is calculated after several propagation (prediction from X_{t-1} to $\overline{X_t}$) and weighting functions (update from $\overline{X_t}$ to $\overline{X_t}'$) and applying a resampling function to the particle set.

C. Control Signal Interpolation

These function applications need to be carried out in the right order with respect to time. Otherwise, no consistent Bayesian state estimation is resembled.

Fig. 3 shows the graphical model of a dynamic Bayesian network which represents multi-sensor integration into a state estimation process. Most importantly, we assume that for each incorporation of a measurement z_i^j there exists an associated state x_i. As a result, the control inputs $u_{\bar{i}}$ need to be translated to the exact time stamps of these z_i^j. Like in the work of [1], we therefore interpolate between different control inputs in order to provide these states.

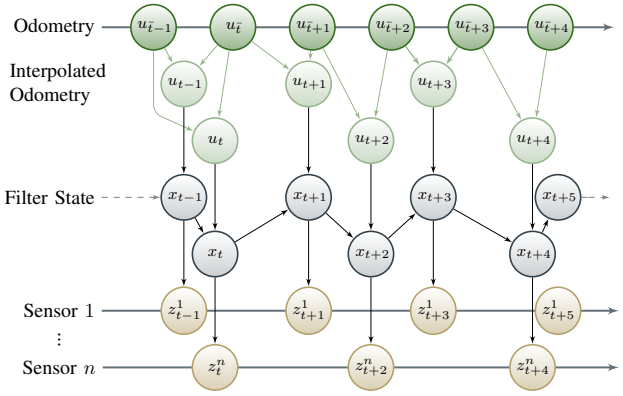


Fig. 3. The dynamic Bayesian network which displays the dependencies of controls, states and measurements in a multi-sensor setting. Sensor readings are displayed in yellow and states in gray. Control inputs are rendered green, while light green indicates inputs which are interpolated in between two original control inputs. In case of the MCL example, the control inputs usually are wheel odometry information.

D. Data Synchronization

To ensure that data from all sensors can be integrated, we also need to synchronize our input streams. For example, stereo data is only valid at its recording time, but is usually delayed by a stereo matching process that produces three-dimensional information from the two retrieved images.

We overcome this problem by calculating the lag of each sensor message as the difference between its recording time stamp, usually provided by driver software, and its time of receipt. Then, we identify the input stream with the largest lag and delay the execution of data from all other input streams such that all input streams share the same lag, i.e. the lag of the most delayed one.

Otherwise, the particles would be propagated further by other sensor inputs, while all delayed sensor messages would be dropped due to their outdated time stamps. Consequently, associated sensors, in our case a stereo camera, would be ignored completely.

What results from this lag propagation method is, that our filter state will be delayed. In order to transfer this state back into the real time domain, the correction between filter state and the last integrated control signal can be estimated and applied to the current control signal. This will be explained in Section V for the 2D MCL instantiation.

E. Scheduling

When combining different types of sensors and different sensor models, we need to take the individual execution times of our models on these data into account.

Inspired by Linux process scheduling, we use *Completely Fair Scheduling (CFS)* [16] to determine the input stream and corresponding sensor model we want to process next. Therefore, we use a priority queue with underlying red-black tree implementation, that guarantees $\mathcal{O}(\log n)$ access time for any operation, to sort our individual model instances by their overall already spent execution time. We execute the one with the least value next, such that each of them is given approximately the same overall execution time. This is illustrated in Fig. 4.

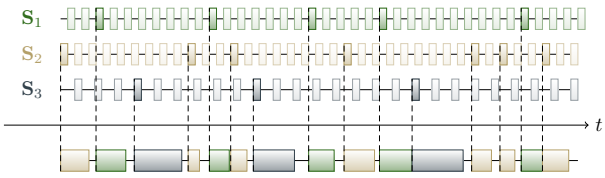


Fig. 4. Three sensors S_1, S_2, S_3 provide data at different rates. The bold sensor messages are processed, the light ones are dropped. The corresponding execution times resulting from the applied sensor model functions are displayed as the length of the associated time slots in the bottom line, which represents the overall execution pipeline.

Whenever a model is executed, its overall execution time is increased by the currently spent execution time, multiplied by a user-defined *nice value*. A low nice value can be used to increase the priority of the associated model.

Further, it would also be possible to employ more sophisticated scheduling strategies, for instance based on the entropy of the different sensor channels.

IV. Framework Structure

An overview of our framework structure, based on the 2D MCL example, is displayed in Fig. 5. The framework itself consists of abstract interfaces for definitions of prediction models P, update models U and resampling schemes R, as well as the SMC filter core.

A. SMC Filter Core

The SMC filter core *SMC* implements the general filter pipeline as defined in Section III-A. Curried prediction and update functions can be passed to queues Q_P and Q_U, respectively, via the functions *SMC::addPrediction* and *SMC::addUpdate*.

In the latter, the lag correction method described in Section III-D is implemented in form of a separate queue for messages that need to be delayed. As soon as a message from the most delayed input stream arrives, the delayed updates are sorted correctly into Q_U. In a main loop, the update queue Q_U is processed, and the prediction queue Q_P is used to predict the filter state at the exact time stamps of the update to be applied.

The updates themselves are passed to a scheduler, which decides if an update shall be dropped or applied to the filter state. The scheduling approach described in Section III-E is implemented in the *CFSRate* class, an instantiation of the interface *Scheduler*.

If the *PredictionIntegral*, where the prediction steps since the last resampling execution are accumulated, exceeds a chosen threshold, the selected resampling scheme is passed to the scheduler. There, conditions e.g. depending on the executed updates can be implemented. For example, our *CFSRate* scheduler applies the resampling step to the filter state only at a fixed rate.

B. Prediction Model

The *PredictionModel* interface can be implemented to model different prediction approaches P. Applying a prediction model at a specific time stamp yields a *PredictionModel::Result* which is a split of the control signal at the exact requested time stamp as well as the model applied to the part contained in the requested time slot. This way, the control signal interpolation mentioned in Section III-C is achieved.

The first part of the control signal is then applied to the particle set and to the *PredictionIntegral*, the other part is integrated back into the prediction queue Q_P.

The *PredictionRelay* class implements the currying function λ_P defined in Section III-B and can be used to pass the curried prediction functions to the SMC filter queue Q_P via the provided function.

C. Update Models

The *UpdateModel* interface can be implemented to model different update functions U^i. The currying function λ_U defined in Section III-B is implemented in the *UpdateRelay* class.

Update models may depend on a *StateSpace* the observation may be compared with, in case of MCL a map m. The *UpdateModel–StateSpace* associations need to be passed to the *UpdateRelay*, such that the desired functions can be generated.

D. Resampling Scheme

The *Resampling* interface can be implemented to model different resampling schemes R. Our framework already provides classes implementing *Multinomial*, *Residual*, *Stratified* and *Systematic* resampling functions as well as a *WheelOfFortune*.

E. Time Domain Decoupling

To avoid interference with the real time domain, the main SMC filter loop with the generated prediction, update and resampling functions is executed in a back end thread, whereas the function bindings and data providing methods are carried out in a front end thread, as displayed in Fig. 5. The decoupling itself is achieved by passing the functions via the two queues Q_P and Q_U.

The SMC filter core communicates the filter state via a *SMCState*, which is an interface that can be implemented to e.g. pass the results to ROS topics.

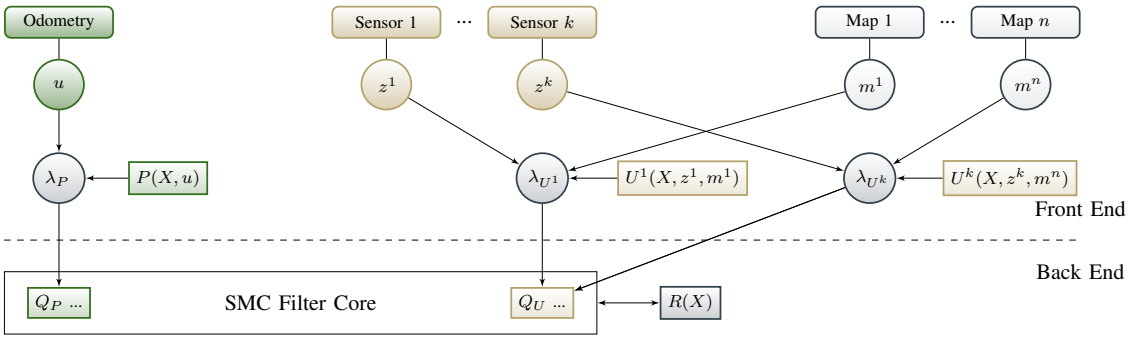


Fig. 5. Structure of our generic Monte Carlo framework, instantiated for 2D Monte Carlo Localization. Control inputs and the propagation model $P(X, u)$ are displayed in green. Sensor readings and the model functions U^i are rendered in yellow. Map inputs m^j are colored gray. The function λ_P is the currying function, which binds the control input u to the propagation model P. The functions λ_{U^i} binds the parameters z^i and m^j to the sensor model functions U^i. These bound functions are passed to the SMC core and stored in a propagation queue Q_P and an update queue Q_U, respectively, where they are sorted by time stamp. The function $R(X)$ is the selected resampling scheme. By design, the acquisition of map and sensor data, as well as binding functions, is contained in the front end of our implementation, while the abstract core algorithm is executed in the back end.

V. Framework Instantiation for 2D MCL

In this section, we explain how we instantiated our abstract framework to implement the 2D Monte Carlo Localization example illustrated in Fig. 5. The implementation we propose is embedded into ROS and provides most implementations as plugins, such that e.g. different sensor models can be exchanged easily at launch time.

A. State Space Domain

In the 2D MCL setting, the state space domain of the SMC method contains relations between the different sensors and the utilized underlying maps. These relations can be represented by transformations between coordinate frames, as shown in Fig. 6.

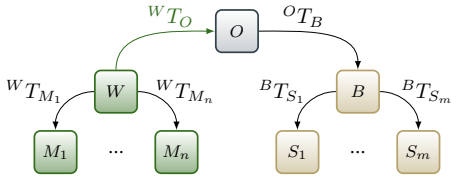


Fig. 6. All maps must be set into relation to a common world frame W, and the sensors fixed to a robot to a common base frame B. Our localization approach then calculates the transformation $^{W}T_{O}$ between W and the robot's global odometry frame O.

To overcome the problem of delayed filter state we already discussed in Section III-D, our 2D MCL instantiation provides the estimated correction $^{W}T_{O}$, which is the transformation between the imprecise robot wheel odometry and the global world frame where the maps are set in relation to. Assuming that the wheel odometry is locally relatively precise, the delay induces only a small drift.

To communicate this transformation as well as the particle filter state in form of 2D poses to ROS, we use a class inherited from *SMCState*.

B. Data Integration

In ROS, sensor data as well as wheel odometry data is communicated via ROS topics. To integrate these data into our MCL instantiation, we implement plugins which listen to these topics and convert incoming data to a common abstract data type. Upon receipt, these data are passed directly to the currying functions λ_P and λ_U implemented in *PredictionRelay* and *UpdateRelay*, respectively, and then the curryied functions are passed to the prediction and update queues Q_P and Q_U of the *SMC* filter.

The maps are pre-loaded at launch time and converted to a common type that inherits the *StateSpace* interface. They are accessible from the *UpdateRelay* via the name of the providing plugin in order to apply currying.

Currently, we provide occupancy grid maps, vector maps and the recently published NDT and ONDT maps [17].

C. Function Definitions

To implement the MCL functions, we provide plugins which wrap sensor models, odometry models and resampling methods. Sensor models inherit the *UpdateModel* interface and odometry models the *PredictionModel* interface.

Besides the well-known likelihood field sensor model and the beam model for occupancy grid maps, which are also integrated in the AMCL localization framework [12], we also implemented the recently published ONDT sensor model [17], as well as the V-AMCL sensor model designed for floor plans [9]. For resampling, we implemented the KLD resampling scheme by Fox [18] in addition to the already mentioned standard methods.

D. ROS Node

Finally, a ROS node loads the different plugins and connects them via the proposed currying functions to a *SMC* filter core. The particles are initialized either via fixed chosen pose or uniform across the maps. The *SMC* main loop is started and stopped via the ROS node.

VI. 2D MCL Experiments

In the following, we present the experiments we conducted to evaluate our proposed concepts. Therefore, we monitored how the different concepts affect the amount of sensor data integrated as well as the localization accuracy of our 2D localization instantiation, compared to AMCL [12] and NDT-MCL [19], whereby we use comparable sensor models.

For evaluation, we used the *Rawseeds* benchmarking suite [20], [21]. The provided indoor datasets contain data from two 2D laser scanners and a stereo camera.

Besides the *absolute trajectory error (ATE)* [22] and the *relative pose error (RPE)* [23], we also evaluate the *drop rate*, where we measure how much of the available data can not be integrated as a sensor update. For instance, the drop rates of the scheduling example in Fig. 4 are $\frac{32}{37} = 86.49\,\%$ for sensor S_1, $\frac{30}{37} = 81.08\,\%$ for sensor S_2 and $\frac{23}{26} = 88.46\,\%$ for sensor S_3. Apparently, these rates roughly correspond with the sensor model runtimes.

A. Lag Correction and CFS Scheduling

First, we evaluated the impact of the proposed CFS scheduling approach as well as the necessity of data synchronization by conducting experiments with and without using the CFS scheduler and with and without using our proposed lag correction (LC) strategy. Without LC, we also did not use CFS, because it would have waited for the most delayed input stream which may never be considered for updates because the filter state may already be further propagated in time. This would have caused a deadlock situation.

In all experiments, we used the *likelihood field* ('LF') sensor model for the *front laser (FL)*, the *beam* ('B') sensor model for the *rear laser (RL)*, both with gridmaps of $5\,\mathrm{cm}$ resolution, and the *ONDT* sensor model [17] for the *stereo camera (SC)* with $20\,\mathrm{cm}$ ONDT maps to investigate different sensor model runtimes. For the basic particle filter and model parameters, we used the AMCL default parameters, which involves e.g. using only 30 points per update to weight each particle and between 100 and 5000 particles. The default parameters are listed on the ROS Wiki page of AMCL [12].

The results can be seen in Tab. I, where the left columns contain the drop rates and the right columns the localization accuracies. All experiments were conducted in real-time.

TABLE I

PERFORMANCE DEPENDING ON THE DIFFERENT MODALITIES

	drop rates			localization accuracy		
	FL [%]	RL [%]	SC [%]	ATE [cm]	T-RPE [cm]	R-RPE [°]
Bicocca-2009-02-25a						
a)	81.92	81.94	**100.0**	5.48 ± 2.72	**1.89**	**0.61**
b)	95.15	95.11	99.82	7.15 ± 4.48	2.14	0.75
c)	92.39	98.44	98.76	$\mathbf{4.45} \pm 3.35$	1.98	0.72
Bicocca-2009-02-25b						
a)	87.16	87.25	**99.99**	5.14 ± 2.89	**2.26**	**0.65**
b)	96.32	96.33	99.79	6.27 ± 3.91	2.48	0.77
c)	92.59	98.47	98.91	$\mathbf{3.95} \pm 2.92$	2.27	0.72
Bicocca-2009-02-26a						
a)	82.01	82.04	**100.0**	5.53 ± 3.16	2.18	**0.67**
b)	95.08	95.10	99.90	6.95 ± 4.02	2.21	0.80
c)	92.31	98.44	98.79	$\mathbf{4.48} \pm 3.66$	**2.09**	0.77
Bicocca-2009-02-26b						
a)	82.13	81.87	**100.0**	5.82 ± 3.31	**2.07**	**0.73**
b)	95.14	95.12	99.83	7.29 ± 6.10	2.60	0.86
c)	92.67	98.60	98.89	$\mathbf{5.24} \pm 5.37$	2.31	0.83
Bicocca-2009-02-27a						
a)	83.41	82.85	**100.0**	8.47 ± 9.33	4.13	**0.67**
b)	95.24	95.16	99.78	10.69 ± 12.02	4.18	0.78
c)	92.56	98.46	98.80	$\mathbf{7.29} \pm 10.32$	**3.16**	0.74

a) wo. CFS, wo. LC | b) wo. CFS, with LC | c) with CFS, with LC

Without lag correction, stereo data is dropped completely or almost completely, as marked bold. This is because the data is delayed by a stereo matching process and therefore outdated and ignored. The best localization accuracy regarding ATE is also achieved with lag correction and with CFS scheduling. Most importantly, our experiments confirmed that lag correction is essential if integrating data from any possibly delayed input stream is desired.

With CFS scheduling, the drop rates seem to correlate with the sensor model runtimes and the sensor frequency. The beam model is clearly slower than the likelihood field model, because ray casting needs to be performed. The ONDT model is also slower than the likelihood field model, because the distributions in the grid cells need to be sampled at the measurement point locations, which is more expensive than a pure likelihood value lookup.

Without CFS scheduling, more of the slow 'RL' updates are executed and the localization accuracy gets worse. Since the stereo camera runs at only approximately $10\,\%$ of the laser scanner frequency, CFS scheduling also leads to lower 'SC' drop rates although the update runtime is high.

For comparison, AMCL with beam model and both laser scanners and without stereo camera achieves a drop rate of about $96.76\,\%$ across all datasets. Overall, with respect to the sensor frequencies, our drop rates are comparable or lower.

B. Localization Accuracy

Finally, we evaluated the overall efficiency of our framework and its 2D MCL instantiation by investigating the achieved localization accuracy of our approach in comparison to AMCL [12] and NDT-MCL [19]. Thereby, we only used the two laser scanners, because AMCL only supports LiDAR data, and applied comparable sensor models with the default parameters of the method to compare against. Further, we used the associated propagation and resampling methods.

The results are listed in Tab. II. Best values are marked bold. In terms of ATE and R-RPE, our framework performs similar or better than AMCL on all datasets. The likelihood field model yields the better results, probably because of its low runtime and therefore more executed updates.

TABLE II

LOCALIZATION ACCURACY WITH LIDAR DATA ONLY

	Model	ATE [cm]	T-RPE [cm]	R-RPE [°]
Bicocca-2009-02-25a				
AMCL	B	4.79 ± 3.48	**1.52**	0.64
MuSe	B	$\mathbf{3.79} \pm 2.79$	1.68	**0.45**
AMCL	LF	5.52 ± 3.77	**1.48**	0.61
MuSe	LF	$\mathbf{2.80} \pm 1.75$	1.61	**0.54**
NDT-MCL	NDT	5.42 ± 4.78	**1.66**	0.60
MuSe	NDT	$\mathbf{4.97} \pm 4.87$	2.01	**0.58**
Bicocca-2009-02-25b				
AMCL	B	4.00 ± 2.41	**1.51**	0.62
MuSe	B	$\mathbf{3.52} \pm 2.26$	1.85	**0.54**
AMCL	LF	4.52 ± 2.69	**1.48**	0.61
MuSe	LF	$\mathbf{3.05} \pm 2.05$	1.71	**0.53**
NDT-MCL	NDT	4.16 ± 4.55	**1.63**	0.58
MuSe	NDT	$\mathbf{3.62} \pm 3.40$	1.91	**0.56**
Bicocca-2009-02-26a				
AMCL	B	4.67 ± 3.15	**1.44**	0.70
MuSe	B	$\mathbf{4.28} \pm 3.32$	1.69	**0.50**
AMCL	LF	5.92 ± 6.32	**1.55**	0.69
MuSe	LF	$\mathbf{3.68} \pm 2.36$	1.59	**0.50**
NDT-MCL	NDT	4.91 ± 4.62	**1.55**	0.63
MuSe	NDT	$\mathbf{4.24} \pm 3.22$	1.85	**0.62**
Bicocca-2009-02-26b				
AMCL	B	4.51 ± 3.20	**1.41**	0.78
MuSe	B	$\mathbf{4.50} \pm 3.61$	1.61	**0.55**
AMCL	LF	5.53 ± 4.42	**1.46**	0.75
MuSe	LF	$\mathbf{4.09} \pm 3.07$	1.60	**0.58**
NDT-MCL	NDT	5.07 ± 4.20	**1.46**	0.68
MuSe	NDT	$\mathbf{4.90} \pm 4.18$	1.77	**0.67**
Bicocca-2009-02-27a				
AMCL	B	6.58 ± 6.12	**1.70**	0.64
MuSe	B	$\mathbf{6.32} \pm 7.30$	2.15	**0.52**
AMCL	LF	7.74 ± 7.72	**1.83**	0.61
MuSe	LF	$\mathbf{5.13} \pm 5.69$	2.07	**0.52**
NDT-MCL	NDT	6.92 ± 7.40	**1.79**	0.59
MuSe	NDT	$\mathbf{6.75} \pm 8.50$	2.18	**0.56**

On the other hand, AMCL results in a better T-RPE. This is most likely related to the MuSe pose being published more frequently, because this step is executed asynchronously and with configurable frequency. Consequently, the pose interpolation required for ground truth alignment gets less smooth, which causes higher RPE values. The same effect also seems to apply in comparison with NDT-MCL.

Consequently, our framework instantiation for 2D Monte Carlo Localization is competitive with other state-of-the-art MCL approaches when using the same functions. Since our framework is plugin-based and already provides a lot of functionality and configurability, it can be used easily for fast prototyping of new functions like prediction models, update models, resampling schemes and scheduling strategies.

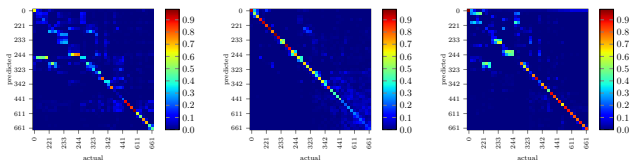
The proposed multi-sensor integration strategies can be disabled and exchanged, respectively, but offer a solid basis for easy integration of any type and amount of sensors.

VII. FURTHER FRAMEWORK APPLICATIONS

The 2D MCL instantiation presented in the preceding Sections was already used for multiple applications. Besides real-world experiments with different robots, it was utilized to localize multiple robots based on a common floor plan in order to collaboratively map their environment in [24].

Further, it was used with plugins that do not pre-load the maps, but dynamically update them at runtime, as a 2D pose tracker in a SLAM framework that supports using multiple map representations [25]. An exemplary excerpt from a pose graph built with LiDAR data from Rawseeds dataset *Bicocca-2009-02-27a* is shown in Fig. 1(b).

Beyond that, a completely different instantiation of the abstract SMC framework was implemented recently to model the problem of contact point localization on a manipulator, as described in detail in [15]. Thereby, the map representation was a 3D mesh, and the particles were located on the mesh edges, which is a completely different state space representation including information about the two neighboring vertices, the 3D position of the particle and its normal to the surface. The propagation step was replaced by a random walk on the mesh surface, and on the resampling step was adapted by additionally spawning uniformly distributed particles. An illustration of this approach can be seen in Fig. 1(a) and some quantitative results are displayed in Fig. 7.



(a) (b) (c)

Fig. 7. Confusion matrices of detected contact points (a) with direct optimization, (b) random forest classification and (c) particle filter localization using MuSe [15]. x- and y-axis denote actual and predicted contact points. Close to the endeffector (high lables, lower right side), the particle filter approach performs best, see [15]. (a), (b) are taken from [26].

Consequently, our framework can be used to model any recursive state estimation problem in any one- or multi-dimensional state space. For the given examples, it simplified and accelerated the development process massively.

VIII. Conclusion

In this work, we presented abstraction techniques and data integration strategies for the problem of integrating different sensor data types into a Sequential Monte Carlo process. This way, we separated the Sequential Monte Carlo core completely from the applied functions and problem space, such that different Sequential Monte Carlo methods can be implemented with our framework rapidly.

The methods were evaluated on the basis of a plugin-based instantiation for 2D multi-sensor Monte Carlo Localization, which is able to outperform state-of-the-art implementations with the same model functions and parameters.

References

[1] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 3923–3929.

[2] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," in *Intelligent Autonomous Systems 13*, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds. Springer, 2016, pp. 335–348.

[3] S. Ito, F. Endres, M. Kuderer, G. D. Tipaldi, C. Stachniss, and W. Burgard, "W-RGB-D: Floor-Plan-Based Indoor Global Localization Using a Depth Camera and WiFi," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 417–422.

[4] R. Miyagusuku, Y. Seow, A. Yamashita, and H. Asama, "Fast and Robust Localization using Laser Rangefinder and WiFi Data," in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Nov 2017, pp. 111–117.

[5] D. Perea, J. Hernández-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta, "MCL with Sensor Fusion Based on a Weighting Mechanism versus a Particle Generation Approach," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 166–171.

[6] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Proc. of the National Conf. on Artificial Intelligence and the Innovative Applications of Artificial Intelligence Conf.*, ser. AAAI '99/IAAI '99. American Association for Artificial Intelligence, 1999, pp. 343–349.

[7] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, and W. Burgard, "Accurate Indoor Localization for RGB-D Smartphones and Tablets given 2D Floor Plans," in *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3138–3143.

[8] J. Biswas and M. Veloso, "Depth Camera Based Indoor Mobile Robot Localization and Navigation," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 1697–1702.

[9] R. Hanten, S. Buck, S. Otte, and A. Zell, "Vector-AMCL: Vector based Adaptive Monte Carlo Localization for Indoor Maps," in *14th Int. Conf. on Intelligent Auton. Systems (IAS)*, Shanghai, CN, Jul. 2016.

[10] P. Elinas and J. J. Little, "σMCL: Monte-Carlo Localization for Mobile Robots with Stereo Vision," in *Robotics: Science and Systems*, 2005.

[11] T. Röfer, T. Laue, and D. Thomas, "Particle-Filter-Based Self-Localization Using Landmarks and Directed Lines," in *RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in AI*. Springer, 2005.

[12] A. H. Brian P. Gerkey, "Adaptive Monte Carlo Localization Implementation." [Online]. Available: http://wiki.ros.org/amcl

[13] A. M. Johansen, "SMCTC: sequential Monte Carlo in C++," *Journal of Statistical Software*, vol. 30, no. 6, pp. 1–41, 2009.

[14] M. F. Fallon, H. Johannsson, and J. J. Leonard, "Efficient Scene Simulation for Robust Monte Carlo Localization using an RGB-D Camera," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1663–1670.

[15] A. Zwiener, R. Hanten, C. Schulz, and A. Zell, "ARMCL: ARM Contact point Localization via Monte Carlo Localization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, (accepted for publication).

[16] I. Molnár. (2007, April) [patch] Modular Scheduler Core and Completely Fair Scheduler [CFS]. https://lwn.net/Articles/230501/.

[17] C. Schulz, R. Hanten, and A. Zell, "Efficient Map Representations for Multi-Dimensional Normal Distributions Transforms," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 2018, pp. 2679–2686.

[18] D. Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *I. J. Robotic Res.*, vol. 22, no. 12, pp. 985–1004, 2003.

[19] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "Normal Distributions Transform Monte-Carlo Localization (NDT-MCL)," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 382–389.

[20] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardos, "RAWSEEDS: Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets," in *Proc. of IROS'06 Workshop on Benchmarks in Robotics Research*, 2006.

[21] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, "Rawseeds Ground Truth Collection Systems for Indoor Self-localization and Mapping," *Auton. Robots*, vol. 27, no. 4, pp. 353–371, Nov. 2009.

[22] RAWSEEDS. (2018, February) Absolute Trajectory Error (ATE). http://www.rawseeds.org/rs/methods/view/9.

[23] ——. (2018, February) Relative Pose Error (RPE). http://www.rawseeds.org/rs/methods/view/11.

[24] C. Schulz, R. Hanten, M. Reisenauer, and A. Zell, "Collaborative Mapping with Pose Uncertainties using different Radio Frequencies and Communication Modules," in *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019, (accepted for publication).

[25] P. Kuhlmann, "Multi-Modal SLAM for Outdoor Robots," Master's thesis, University of Tübingen, 2019.

[26] A. Zwiener, C. Geckeler, and A. Zell, "Contact Point Localization for Articulated Manipulators with Proprioceptive Sensors and Machine Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 323–329.