Creating a robot localization monitor using particle filter and machine learning approaches

Matthias Eder¹ · Michael Reip² · Gerald Steinbauer¹

Accepted: 16 December 2020 / Published online: 12 February 2021
© The Author(s) 2021

Abstract

Robot localization is a fundamental capability of all mobile robots. Because of uncertainties in acting and sensing, and environmental factors such as people flocking around robots, there is always the risk that a robot loses its localization. Very often behaviors of robots rely on a reliable position estimation. Thus, for dependability of robot systems it is of great interest for the system to know the state of its localization component. In this paper we present an approach that allows a robot to assess if the localization is still correct. The approach assumes that the underlying localization approach is based on a particle filter. We use deep learning to identify temporal patterns in the particles in the case of losing/lost localization. These patterns are then combined with weak classifiers from the particle set and sensor perception for boosted learning of a localization estimator. Through the extraction of features generated by neural networks and its usage for training strong classifiers, the robots localization accuracy can be estimated. The approach is evaluated in a simulated transport robot environment where a degraded localization is provoked by disturbances caused by dynamic obstacles. Results show that it is possible to monitor the robots localization accuracy using convolutional as well as recurrent neural networks. The additional boosting using Adaboost also yields an increase in training accuracy. Thus, this paper directly contributes to the verification of localization performance.

Keywords Robot localization · Neural network · Machine learning · Particle filter

1 Introduction

Robot localization capability of all mobile intelligent robots, aiming to determine the position and orientation of an object in space, which is also called an objects pose. There is a huge corpus of research that has been

conducted to allow reliable estimation of the current pose using uncertain sensor data and action execution. Typically, these approaches perform some kind of state estimation to determine the current robot pose, using sensors like 2D lasers [1]. A typical approach used for localization are particle filters [2]. Particle filters are a popular method for representing arbitrary probability distributions and solving state estimation problems. The technique behind particle filters is the Monte Carlo method [3–5] which has existed for over five decades. The key idea of particle filters is to spread particles in space which represent the posterior distribution [1]. Instead of using a parametric form for representing the distribution, particle filters generate samples based on its own distribution. Figure 1 shows a general overview on the localization with a particle filter. First, an environment must be mapped. This is often done using a simultaneous localization and mapping (SLAM) approach [6], which needs takes the robot movement (twist) and scanning data on the environment into consideration. The result is a map of the environment that can be used in addition with scanner data and the robot's twist to estimate the pose of the robot.

This article belongs to the Topical Collection: *Special issue on Artificial intelligence in practice - from theory to application*
Guest Editors: Franz Wotawa, Gerhard Friedrich and Ingo Pill

✉ Matthias Eder
matthias.eder@ist.tugraz.at

Michael Reip
m.reip@incubedit.com

Gerald Steinbauer
steinbauer@uist.tugraz.at

¹ Institute of Software Technology, Graz University of Technology, Graz, Austria

² IncubedIT GmbH, Hart bei Graz, Austria

Fig. 1 General overview on localization using particle filters

The main problem with particle filters is that they are computationally expensive. Particle filters allow the analysis of complex systems which are non-linear and non-Gaussian. The goal is to deal with arbitrary probability distributions and model them correctly [7, 8]. Since the computational power has increased in the last years and particle filters are a non-parametric approach for solving complex models, they were applied in many different fields like neuroscience [9], biochemical networks [10], signal processing [11], economics [12] and robotics [13].

Because of uncertainties in sensor data and environmental factors such as people flocking around robots there is always the risk that the error of the pose estimation increases significantly or that it diverges completely. Very often navigation and other behaviors of robots rely on a valid pose estimation. Thus, for performance as well as dependability of robot systems it is of great interest for the system to know the state of its localization component. Moreover, if a robot gets delocalized near an object this can result in undesirable behaviors, such as a crash. To avoid this, it is aimed to detect the delocalization before it occurs to be able to perform a predefined error-correcting behavior. Thus undesired behaviors can be prevented.

In this paper we present an approach that allows a robot to assess if the localization is still valid and thus contribute to the topic of localization performance verification. The approach assumes that the underlying localization approach is based on a particle filter. Following the assumption that the particle set representing a probability distribution of the robot's location bears information about the localization state, we propose to use deep learning to identify temporal patterns in the particle set to be able to act in pre-caution before losing localization. Thus, we investigated different deep neural networks (non-recurrent and recurrent) for their ability to learn identifying such patterns. Moreover, we propose to combine these networks with weak classifiers obtained from statistical information in the particle set and the actual robot perception for boosted learning of a more reliable localization monitor.

For the training and evaluation of the approach we use a realistic simulation of an industrial transport robot environment where due to access to ground truth a large amount of labeled data can be generated. Moreover,

a simulation allows to provoke a degraded localization easily by disturbing the robot's localization system by randomly adding dynamic obstacles. By disturbing the sensor readings, the localization algorithm cannot fit the sensor data to the prerecorded map and thus might delocalize the robot.

The remainder of the paper is organized as follows. In the next section we discuss briefly related research. In Section 3 we introduce the proposed approach for a localization monitor. An experimental evaluation of the proposed approach is presented in the succeeding section. Finally, in Section 5 we draw some conclusions and provide some ideas for future work.

2 Related research

In this section we briefly discuss related research concerning robot localization and machine learning. First, problems of current particle filter approaches are presented before discussing recent advances in the topics of robot localization and machine learning.

2.1 Dilemma of particle filters and its limitations in robot localization

Localization approach based on particle filters are a good method to overcome common issues in the field of robot localization [1, 2]. For example they are able to solve the global localization problem as well as the kidnapping problem. Another advantage of particle filters is that they are non-parametric [1] which means that they get an arbitrary probability density function (PDF) without adjusting parameters by hand. However, there also occur some minor problems. The first is that the accuracy and the performance depend on the number of particles used to represent the PDF. To keep this algorithm efficient the number of particles should be small while to keep it accurate the number of particles should be high [14, 15].

Another issue is the inflation of the particle set. When sensor data can not be clearly assigned to a pose within the environment, the particle set starts to inflate because the uncertainty about the current pose increases. It is

hard to determine if the robot is still localized or if it is already delocalized. The best example for such an inflation are a long straight corridor. When navigating through the corridor the particle filters pose estimation of the robot becomes imprecise. This is due to the issue that for the particle filter every pose within this corridor looks similar. The current laser scan fits fine to the current robots pose but also fits to various poses ahead and behind the robot due to ambiguous measurements. This paper aims to solve this inflation problem by using the proposed method and detect uncertainty increases. This way, an impending delocalization can be detected in advance and the robot can start a predefined error-correcting behaviour.

2.2 Recent approaches in robot localization and its accuracy

A publication which was presented in 2012 by Röwekämper et al. evaluates the position accuracy of a mobile robot localization method that is based on particle filtering and laser scan matching [16]. For evaluation they used a motion capture system that tracks the position of a robot within its environment with high accuracy [17]. The results show, that a high accuracy of $\pm 0.5\text{cm}$ can be achieved in a static environment. The localization system which was used in their evaluation was a combination of basic state-of-the-art approaches. Therefore Monte Carlo Localization, a scan matching procedure [18] and the Kullback-Leibler distance sampling (KLD-sampling) [15] were used. Although the proposed methods yields a high localization accuracy, the combination of particle filter and laser scan matching have the potential the delocalize the robot in a dynamic environment. To overcome this, a similar approach is used in this paper in combination with the presented approach.

A novel approach is proposed by Kallasi et al. who introduced a new method for detecting features to localize and navigate robots [19]. They propose two new feature detectors called *Fast Adaptive Laser Keypoint Orientation-Invariant* (FALKO) and *Orthogonal Corner* (OC). Those two detectors are an improvement of the *Fast Laser Interest Region Transform* (FLIRT) [20] approach that can be used to detect high curvature points in laser scan images. While the FLIRT method searches for general features which depend on the viewpoint of the robot, FALKO and OC are designed to detect stable features like corner walls. The difference between the two proposed methods is that FALKO detects features by selecting meaningful neighbours and scoring the cornerness of a feature and OC uses orthogonal alignments to detect important features.

Since particle filters are among the most popular methods for state estimation, a lot of research is done to improve their accuracy and performance. In general the state of a complex system can be estimated correctly by using (infinite) many

particles. But with an increasing number of particles the efficiency of the particle filter decreases. Thus, researcher aim at decreasing the number of particles while offering the same performance. One attempt is called adaptive particle filtering as described in [13].

Oliveira et al. present a visual localization approach to estimate a robots pose [21]. In contrast to Lidar systems, visual pose estimation is more cost-effective but the accuracy is usually inferior to Lidar systems. To overcome this issue, Oliveira et al. present a vision-based localization approach that uses visual odometry and topological localization which is used to train deep neural networks. Using their approach, they manage to get up to 10 times smaller localization errors than with traditional vision-based localization approaches.

Havangi recently proposed a localization method, using particle swarm optimization (PSO) estimators to overcome the localization weaknesses which occur when using particle filters [22]. In the proposed method, dynamic optimization is converted to find a pose estimate that suits best to the robots location. The advantage compared to PF is, that no noise distribution and resampling step is required.

A different localization approach is the use of ultrawideband (UWB) sensors, such as proposed in [23]. In this paper, Güler et al. propose an on-board multi-robot localization approach which does not require infrastructure. therefore, three UWB sensors are mounted on an anchor robot which estimates the others robot location without inter-robot communication. However, the usage of UWB for localization has the major disadvantage that it is usually not very accurate, especially then obstacles are within the way.

Other recent localization monitoring approaches can be found in [24, 25]. In there, integrity risk metrics and methodologies are presented to verify a robots localization performance.

2.3 Review on machine learning approaches

The main application of convolutional neural networks (CNN) is the detection of features within images. Therefore a given image is used as input of a neural network and sent into convolutional layers which store single features in feature maps. Those features can then be used to recognize patterns for classification.

A simple and modern example is presented by Shamov and Shelest [26]. They present the main features of convolutional neural networks and show how they are applied for feature detection. For this purpose they created the task of detecting tower lighthouses from a video stream.

A new application of convolutional neural networks is the detection of dynamic obstacles in grid maps. Piewak et al. use a grid map and a deep neural network to detect whether grid cells within a map are moving [27]. The difference to a

normal tracking approach, like particle filters, is to use the complete map grid with a top view as input image. They also proposed an approach which is optimized for real time applications.

A Long-Short Term Memory (LSTM) network is good at classifying, predicting and processing time series. Its special power is that it is also good in learning and evaluating time series which have a time lag of unknown size between two events. It is well suited for training data that really has to remember longer time series and also focuses on the current input. This is a reason why in many fields LSTM performs better than other recurrent architectures [28]. Gensler et al. investigated how good LSTM networks can predict a future event [29]. Therefore they compared different types of deep learning networks on a defined task. The task was to forecast the power production of solar panels. With this task they want to provide a reliable power forecasting method that allows to efficiently operate a solar power station. The results of the experiment for solar power forecasting showed that in nearly all error measurements an auto-encoded LSTM performed the best. This also shows that recurrent networks and especially long-short term memory networks excel at their task of predicting future events.

Pattern recognition is an important task in visual computing. It is about detecting specific patterns within an given input. In visual computing these inputs are images which are used to detect certain content like humans. Modern pattern recognition methods are based on convolutional neural networks which aim to identify patterns within the input data by training the network on a big set of data. This method can be used to detect licence plates on a car [30] and is also applied in vineyards to detect birds which needs to be scared away [31].

Morales et al. presented a method for object tracking by using a 3D occupancy grid as environment representation and a particle filter based approach for detecting and tracking obstacles [32].

In [33], a classification approach is presented to detect the invasiveness of pulmonary subsolid nodules in CT images. Although the topic is not in direct relation with robot localization, the training approach using adaptive-boost deep learning shows how to train a strong classifier using weak CNN-based classifiers. The results show, that it is possible to achieve an acceptable accuracy when training features extracted from a CNN for a binary classifier.

3 Localization accuracy estimation approach

In order to judge if the estimation of the robot's position is accurate we follow the approach depicted in Fig. 2. In our application we need to solve a binary classification problem: either the robot is well localized or delocalized. We declare the robot delocalized if the estimated position and orientation diverged form the true position and orientation more than a given threshold.

In order to have access to a ground truth of the position and orientation and to be able to provoke delocalization we simulate the robot's motion and sensing (i.e. laser scans) as well as the environment. The data from the perception as well as the data from the particle filter-based pose estimation (i.e. particles) are recorded and converted to labeled training data for the following classification steps. For the classification we follow two approaches which we also use in combination. The first approach follows the assumption that the probability distribution of the robot's pose modelled by the particle set and its typical growth of uncertainty bears information about the localization accuracy. Thus, we train non-recurrent and recurrent neural networks for the classification based on the particle set. The second approach follows the assumption that statistical features (weak classifiers) derived from the perception and the particle set can be used to learn a strong classifier using boosting, since the particle set represents a distribution

Fig. 2 The overall approach for estimating the accuracy of robot localization

in the environment. By analyzing this distribution with statistical tools and using the received scan data to match it to the environment, information on the localization state might be revealed. To see, how promising a feature is, the KL-divergence between a localized and delocalized set is calculated, as described in Section 3.3. Finally, the classification from the trained neural network can be used in addition to the weak classifiers in a boosting step, to see if the neural network output can further improve the classifier.

In the remainder of the section we introduce the following steps of the proposed process in more detail: (1) generation of training data, (2) selection and validation of neural network architectures, (3) validation and selection of features, and (4) learning classifiers.

3.1 Generation of training data

This section describes how the training data for the neural networks is generated.

Figure 3 shows the overall procedure of generating a training sample and finding the correct label. To create a sample one needs the particle set from the particle filter which is used to create a training image x . A particle represents a potential pose (position and orientation) along with a weight representing the importance of the particle $\langle x, y, \theta, w \rangle$. As the performance of the particle filter depends on the size of the particle set, a sufficient number of particles has to be selected. In this work, a particle set size of $M = 1000$ was selected. We then convert the particle set into a binary image because it is able to represent the distribution of the particle set and is suitable for state-of-the-art machine learning tools. The image itself then contains black pixels which represent a particle of the particle cloud. The image is then labelled by comparing the exact pose $\langle x_{GT}, y_{GT}, \theta_{GT} \rangle$ (retrieved from the simulation) and the estimated pose $\langle x_{PF}, y_{PF}, \theta_{PF} \rangle$ (retrieved by the distribution of the particles). Using a defined distance α, β as threshold between them one can classify the robots localization state and thus also create a label y for the

training sample. the values for the thresholds are defined in Section 4.2.

$$y = \begin{cases} 0, & \text{if } \sqrt{(x_{GT} - x_{PF})^2 + (y_{GT} - y_{PF})^2} < \alpha \wedge \|\theta_{GT} - \theta_{PF}\| < \beta \\ 1, & \text{else.} \end{cases} \quad (1)$$

Usually the particles are distributed over a large state space (i.e. the entire environment map). Since this map does differ in size and the particles accumulate at one area, it is not necessary to store the complete map as data sample. In order to focus on relevant areas of the particle distribution, the image is centered around the mean of the positions represented in the particle set. Moreover, the image is oriented along the mean of the orientations represented in the particles. Then the area of the distribution is cut and cropped into a default size. By analysing the distribution of the particle, the optimal area size was estimated. This was done by cutting the the area to a size $s \times s$ meters that when plotting the particle set, in 95% of the cases all particles are in the specified area. To determine the optimal side, sample data was recorded which collected the information of the particle cloud. The result was a cutting area with $s = 1.5m$. Using only a focused representation has the advantage that the trained network can be used for environments of different size and shape. The cut area is then cropped into an image of size $p \times p$, where p is the number of pixels on the axis. The size of p depends on the details which are represented by the particle distribution and on the desired performance of the neural network training. After analyzing different values for p by pre training small neural networks, a size of $p = 36$ pixel was selected as this showed the best results with acceptable training performance. This values had been selected based on the results of run-time and accuracy.

Figure 4 shows generated images which are created during the movement of the robot, stored at different time steps. The first three images illustrate a particle set of a

Fig. 3 The procedure of generating a data sample and labelling for training

Fig. 4 Example images generated for training the neural network. **a** Localized particle set at time step t . **b** Localized set at time $t + 40$ s. **c** Localized set at time step $t + 90$ s. **d** Delocalized image at time step $t + 110$ s

well localized robot while the fourth image is an example of a delocalized robot. To provoke the delocalization of the robot, dynamic obstacles were included, so the particle filter algorithm got disturbed.

The transformation and inflation of the particle cluster are indicators for the localization state of the robot. Thus a neural network can be trained to detect relevant patterns within this transformation.

To generate enough distinctive data samples the robot has to drive randomly within a predefined environment. This leads to different robot locations within the environment and samples representing different localization situation in the environment. Figure 6a shows an example environment. Due to the facts that methods based on particle filters are quite robust and we need positive as well as negative examples we need to provoke delocalization. One of the simplest and most realistic attempts is to disturb the lidar sensor using dynamic obstacles that are not represented in the map. For this purpose we randomly generate additional obstacles around the robot in the simulation. Such obstacles cause unexpected measurements and may lead to delocalization.

3.2 Selecting and validating the neural networks

Having created a training set for neural networks one has to define an appropriate network structure (type, layout) which can be used for localization classification. Thus, several possible network structures need to be trained and validated.

When using data from a particle filter as input a priori it is hard to specify the best network type and structure since it is rather unclear which information is extracted by a neural network for estimating the robot's localization accuracy. A possible network structure is a convolutional neural network (CNN) which is used for pattern recognition. Another possibility is a recurrent network that may learn the transformation of the particle set over a certain time period.

To evaluate a wide range of possibilities, both network types are considered. A CNN for feature extraction and a recurrent network structure for learning the transformation

over time. For learning a recurrent network a long-short term memory (LSTM) structure is used since a better performance is expected [28]. Additionally a combination of both, CNN and LSTM, is trained. This is called a Long-Term Recurrent Convolutional Network (LRCN). By using this network structure it is evaluated if the advantages of both former types can be combined.

For each different network types (CNN, LSTM and LRCN) three different network layouts are created and trained. We generate a simple, a moderate, and a complex layout. Those three layouts are chosen to evaluate whether a simple network structure leads to underfitting or a complex structure leads to overfitting. This results in a total number of nine networks to be trained, which are summarized in Table 1.

To validate the usefulness of a network structure one needs to define a metric that allows to compare different network structures in respect to localization monitoring. A suitable metric is accuracy [34]. Accuracy is a value that describes how well a given data set is classified. If a classification task with two classes is considered it can be used to calculate the accuracy by dividing the predictions of a data sample into a binary classification.

Having two classes, one can assign to a class either the *positive* or *negative*. To validate a data sample it is fed into, the network structure and the output is recorded. The output is then assigned to one of the two classes, depending on the result. By comparing the assigned class with the expected class label one can determine whether a class was correctly assigned. If both, the result and the label match, then the outcome is *true*. When a data sample is classified as positive and the expected class is also positive the data sample is *true positive*. Vice versa, if the data sample is classified as negative and the result is negative, it is categorized as *true negative*. If the two classifications do not match the outcome is *false*. If the data sample is classified as negative and the expected result is positive the outcome is *false negative* and *false positive* if the data sample is labelled as positive and the prediction is negative. By counting the results for the

Table 1 Overview of the tested network types and their complexity

NN Type	Complexity	# Convolutional layers	# LSTM layers	# Other layers	# Total layers
CNN	simple	1	-	2	3
CNN	mid-complex	2	-	4	6
CNN	complex	21	-	11	32
LSTM	simple	-	1	3	3
LSTM	mid-complex	-	2	3	5
LSTM	complex	-	3	4	7
LRCN	simple	2	1	2	5
LRCN	mid-complex	3	2	5	10
LRCN	complex	5	2	7	14

examples in the data set one can determine the accuracy as

$$acc = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \quad (2)$$

where t_x indicates the number of correct classification and f_x the number of the incorrect ones with x either p (positive) or n (negative).

3.3 Feature selection

Once an appropriate network structure has been selected useful features for the boosting step needs to be identified. By using features which can be extracted from the particle set and use them in a boosting step, it is aimed to improve the accuracy for estimating the localization state of the robot. In the context of the company in which the project was carried out, a localization state estimation called localization scoring was used prior this implementation. This localization scoring approach uses features that are extracted by comparing the scan data to the robots map. For example a Hough line transform [35] was performed on the map and scan points and compared to each other. Using features form the existing localization scoring mechanism and expanding it with relevant data on the particle set, such as the center of gravity, the features were investigated to see whether they reveal information on the robots localization estimate.

In general it is hard to specify which features may be important for localization scoring. Using a simple evaluation one can search for various possible features that might be useful as weak classifiers in scoring the accuracy of a robots localization. By using the thresholds α and β as above the robots localization status can be easily determined and every feature can be inspected on its own. We identified 33 possible features that are based on information from the robot's perception and data from the particle filter. Table 2 summarizes the selection of these features. For a complete description of the list we refer to [36]. During the recording

of the training data feature estimations are recorded and labelled with their localization state, as described in (1).

To determine which features are important the robot was randomly driven around in a simulation environment and the features were recorded. After generating roughly 100.000 feature samples every feature is analysed individually. These recordings are split into a localized X_Q and delocalized X_P data set. These sets are then represented as a discrete distribution over k bins to determine the divergence between the two classes. However, as continuous variables reveal more information about the robots location estimate, the original continuous values of the were used again for boosting after selecting the relevant features. The size k of the bin is selected by observing the prerecorded data set and evaluating its maximum and minimum values. Then, a preselected bin size was chosen considering the unit. for meters the bin size was 1cm, for percentage the bin size was 1% and for degree the bin size was 1°. Each bin contains a number of samples which lie within that bin. A discrete probability distribution P is then represented as the number of samples within that bin divided by the total number of samples n

$$P(i) = \frac{|bin(i)|}{n} \quad (3)$$

The same holds for the distribution Q . Having two discrete distributions P , Q for the localized and delocalized set one can calculate the Kullback-Leibler divergence [37] as

$$D(P||Q) = KL(P, Q) = \sum_i^k P(i) \times \log \frac{P(i)}{Q(i)} \quad (4)$$

where k is the number of bins. The KL-divergence is only defined if $\forall i : Q(i) = 0 \rightarrow P(i) = 0$ applies. If $P(i) = 0$ the contribution of the i -th bin is also 0. It is defined that $D(P||Q) \geq 0$ for all distributions and $D(P||Q) = 0$ if $P = Q$.

Using the KL-divergence between the probability distribution of the feature values for the positive and negative

Table 2 List of features which were used as weak classifiers

Feature	Unit	#bin	$D(P Q)$
(1) Center of gravity maximum distance	meters	350	0.265
(2) Center of gravity mean	meters	150	0.313
(3) Center of gravity mean absolute deviation	meters	100	0.314
(4) Center of gravity median	meters	150	0.296
(5) Center of gravity median absolute deviation	meters	50	0.321
(6) Center of gravity minimum distance	meters	100	0.153
(7) Center of gravity standard deviation	meters	100	0.293
(8) Circle maximum distance	meters	300	0.286
(9) Circle mean	meters	200	0.294
(10) Circle mean absolute deviation	meters	50	0.281
(11) Circle median	meters	200	0.279
(12) Circle median absolute deviation	meters	50	0.260
(13) Circle minimum distance	meters	150	0.159
(14) Circle standard deviation	meters	50	0.285
(15) Point distance	meters	400	0.086
(16) Point fitting	%	100	0.157
(17) Point inlier	%	100	0.118
(18) Point quality	%	100	0.035
(19) Raycasting inlier	%	100	0.183
(20) Raycasting inlier percentage	%	100	0.046
(21) Raycasting matching percentage	%	100	0.166
(22) Raycasting outlier percentage	%	100	0.045
(23) Raycasting quality	%	100	0.098
(24) Angle inliers	%	100	0.315
(25) Angle quality	%	100	0.035
(26) Line angle	degree	360	0.105
(27) Line distance	meters	400	0.042
(28) Line fitting	%	100	0.033
(29) Line length	meters	250	0.086
(30) Number of clusters	amount	5	0.102
(31) vmain cluster variance x	meters	100	0.403
(32) main cluster variance y	meters	100	0.050
(33) main cluster variance z	meters	150	0.147

Grey rows indicate discriminating features while white rows indicate non-discriminating features

cases allows to select well discriminating features. The higher the KL-divergence the more different the distributions are and the more informative is the underlying feature. On this basis, promising features can be selected for the boosting step. As criterion for the feature selection a threshold of $D(P||Q) \geq 0.1$ was selected.

Table 2 shows the KL-divergence of some example features calculated using the same simulated disturbed navigation as discussed above. The selected features which were used for boosting are highlighted in grey. Figure 5 depicts the probability distribution of feature values for a discriminating and a non-discriminating feature.

3.4 Adaptive boosting

Adaptive Boosting (AdaBoost) is a machine learning approach which uses supervised data for classification. Since a single feature by itself is a rather bad classifier for estimating the localization state, the idea is to combine multiple weak classifier which do not provide enough information about a certain class itself and combine them into a strong classifier that can be used for identifying classes [38, 39]. In general an adaptive boost algorithm takes N training samples (x_i, y_i) , $1 \leq i \leq N$ with $x_i \in \mathbb{R}^K$ and $y_i \in \{-1, +1\}$. x_i is the input vector which contains

Fig. 5 Probability distributions of feature values for localized and delocalized cases for feature 18 (left, non-discriminating) and feature 31 (right, discriminating)

K different components (features) that are used for training. y_i is the desired label which is either -1 or $+1$. There exist several variants of boosting algorithm which all have a similar structure [40]. In this work the standard discrete Ada Boosting algorithm which uses two classes is used. It uses its N -sized input set and initializes the weights for each input sample with $w_i = 1/N$. Then a weak classifier $f_m(x)$, the weighted training error ϵ_m , and the scaling factor c_m is computed. For a detailed formula on how to compute these values, we refer to [41]. Then the weights are increased for input samples that have been wrongly classified. After this step the weights are normalized and the steps for finding a new weak classifier are repeated M times. At the end a final classifier $F(x)$ is found which uses the sign of the weighted sum of the input set. This can then be used to estimate the state of the input data, in our case the localization state.

In the proposed approach we use an appropriate selection of features from the previous section and the same examples from training trajectory used in training the neural network. Moreover, we treat the binary classification of the trained neural networks as additional weak classifier.

3.5 Support vector machines

A Support Vector Machine (SVM) is a popular tool for classification problems [42, 43]. It searches for an optimal hyperplane which can be used to separate two classes. It takes training samples as input which are classified and outputs an optimal hyperplane which can be used to categorize new samples. To find the optimal plane which separates the two classes best is not a trivial task. Consider m training samples $(\mathbf{x}^{(i)}, y^{(i)})$, $1 \leq i \leq m$ where $\mathbf{x}^{(i)}$ is a sample which is labelled with $y^{(i)} \in \{-1, 1\}$. If those samples are linearly separable, multiple possible solutions exist which could be applied. The task is now to find the best hyperplane, determined with $\mathbf{w}_o$ and b_o , that maximizes the separation space between the two classes $\mathbf{w}_o^T \mathbf{x} + b_o = 0$. If data samples are not linearly separable, the so-called kernel methods can be applied. Those methods take the training samples which are not linearly separable and map

them into a higher dimensional space using some non-linear transformation $\varphi(x)$ [44]. Kernels allow to operate in high-dimensional feature spaces and allow a linear separation of the classes within a different state space. This is also called the kernel trick and has better performance than explicit computation methods.

As for the approach with AdaBoost, we use the features selected above in the previous section and the binary classification result from the neural networks as classifiers for the SVM. At the end, the two methods, AdaBoost and SVM, are then to be compared to each. First, the extracted features from Section 3.3 are used to estimate the localization state of the robot. Then, the binary classification of the trained neural network is used as additional feature to see, whether this further improves the performance of the SVM or Adaptive Boost.

4 Experimental evaluation

In the experimental evaluation we are interested to investigate how well the proposed approaches are able to identify a lost localization of a robot. Moreover, we are interested which network structure minimizes the uncertainty and if the approach is dependent on the environmental structure. In the remainder of the section we introduce the experimental setup, give details on the preparation of the training data, and discuss the results with respect to the used network structure and the used features as well as the boosting step.

4.1 Experimental setup

The training and evaluation is based on a simulation of a team of transport robots in industrial environments based on Stage [45] and provided by an industrial partner. The used navigation software is the same as embedded into the real transport robots and uses laser scans, odometry, a gyro, and a 2D gridmap for the localization based on a particle filter. The simulation is based on ROS Indigo [46] and runs on

Fig. 6 Environment maps used for training and evaluation. Based on an real industrial environment enriched with additional clutter

a standard PC equipped with an Intel i5 (2.5 GHz), 4 GB RAM, and Ubuntu 14.04.

The training and classification is based on the libraries Caffe 1.0.0, OpenCV 3.3, and CUDA 7.5.17 and runs on a standard PC equipped with an Intel i7 (2.1 GHz), 8 GB RAM, Nvidia GPU, and Ubuntu 16.04.

Figure 6 depicts the environment map used where the left map was used to collect data for training and the basic evaluation and the right map was used to evaluate if the trained classifiers generalize to other environments.

4.2 Training data

As described in Section 3 the robot is randomly driven through the training environment which is simulated. As stated earlier, the simulated environment allows to add random obstacles within the environment that disturb the scan data and thus provoke delocalization. Also the precision of the scan data can be defined. In our simulation environment, the precision of $\pm 3\text{ cm}$ are used, as this is the standard error of currently existing sensor systems. During this run all relevant output from the particle filter and the perception were recorded. Using the thresholds $\alpha = 1.2\text{ m}$ and $\beta = 1.2\text{ rad}$ all data points were labeled. Localized samples got the label 1 and delocalized samples the label 0. The thresholds were selected in the way that they maximize the KL-distance of the hand-crafted localization scoring approach.

This leads to a sequence of 150.000 data points where 50% are labeled delocalized. For the training of the non-recurrent networks this data are equally split into a training and evaluation set and shuffled randomly. For the training of the recurrent networks individual data points are useless.

Thus, we split the data points in positive and negative sequences of the length of 2×7 data samples. The negative sequences are organized around a time point t where the robot become delocalized and 7 data samples from before and after t are added. Using a frequency of 5Hz this results in a sequence of about 3 seconds. This lead to 10.714 sequences. Moreover, the same number of positive sequences were generated where the robot stayed localized. Table 3 gives an overview on how the data was separated for training, validation and test purposes. The same data sets were also generated for the second environment which are used to evaluate the generalization of the approach and not used in training.

4.3 Validation of the neural networks

In order to evaluate the most appropriate network structure for the estimation of localization accuracy we trained each network structure with the proper (individual data points or sequence of data points) training data and evaluated the results using the evaluation set of the first environment as well as the data set of the second environment. As criterion we use the metric defined in Section 3.2.

Table 4 summarizes the results of this evaluation. The number of iterations in training was 100.000 for all networks. In general it can be seen that all networks perform quite well on the training data where we have equally many positive and negative examples. The mid-complex CNN performs best. For the validation set in the same environment the performance drops by about 2-13%. The mid-complex CNN is still in the lead but shows the largest performance loss showing a tendency for overfitting. For the data set from novel environment the performance drops for

Table 3 Separation of training data, validation data and test data

Label (0/1)	Training dataset		Validation dataset		Test dataset	
	1	0	1	0	1	0
Non-recurrent (# data samples)	56.250	56.250	18.750	18.750	75.000	75.000
Recurrent (# of sequences)	8.035	8.035	2.679	2.679	10.714	10.714

The number of data samples is equally distributed over localized (label 1) and delocalized (label 0) samples

Table 4 Result for all network structures: simple CNN (A), mid-complex CNN (B), complex CNN (C), simple LSTM (D), mid-complex LSTM (E), complex LSTM (F), simple LRCN (G), mid-complex LRCN (H), complex LRCN (I)

Environment	Accuracy/%								
	A	B	C	D	E	F	G	H	I
Training set	84.52	96.47	91.22	84.42	86.95	87.42	88.73	86.12	86.09
Validation set	79.62	83.25	81.38	82.16	81.99	83.10	82.55	81.34	79.30
Test set	67.63	66.27	62.06	67.79	67.36	66.33	69.55	67.19	58.88

all networks to around 67%. The leader is here the combined simple network showing that this network structure is able to generalize well.

The accuracy is obviously a compressed metric. In order to analyze the performance of the mid-complex CNN, Table 5 shows the detailed results of its evaluation. It can be seen that the true and false results for the positive and negative cases are almost balanced. Thus, in contract to other networks this network is not biased in a sense of being too optimistic (many false negatives - missing delocalizations) or too pessimistic (many false positives - raising a lot of false alarms). Figure 7 shows the loss function during the training of the mid complex network. It can be seen, that the loss is decreased within the first 60.000 iterations and then does not decrease further. Thus, a total of 60.000 iterations would have been sufficient to train this neural network. We refer to [36] for more detailed results.

4.4 Result of SVM

The first method for classifying the localization state of a robot with various features are Support Vector Machines. SVMs are used in this paper to find an optimal separation hyperplane that separates the features introduced in Section 3.3 into localized and delocalized states. Therefore, two different feature sets are used for training. At first the selected features from Section 3.3 are used without the binary classification of the selected neural network output. This is then compared with a SVM that trains all the features including the neural network output. For finding an optimal separation hyperplane the library offered by OpenCV is used. This library offers an automatic trainer that automatically searches for the optimal separation hyperplane and also tries to detect the perfect Kernel for it. Since this SVM algorithm is well

tested by OpenCV and its automatic mode decreases the computational effort this method was used for training.

Table 6a shows the outcome of the boosting with a SVM. It appears that this method is not applicable for this task since the testing accuracy in the same environment is with 61.94% lower than the results of the neural networks themselves. The accuracy in a different environment decreases further to 58.11% which is shockingly close to a random distribution of 50%. However, since this is the base line for comparison the neural network feature is still added and evaluated with a Support Vector Machine to see if it can be improved too. By adding the neural network feature one can improve the accuracy of a SVM in the same environment by about 2% as shown in Table 6b. Although this is an improvement compared to the SVM without the network feature it is still an insufficient result. Also, it can be seen, that the SVM fitted the network to detect more false negatives than false positives within the training environment. Due to the low accuracy it can be assumed that a SVM is not suitable for solving the task of estimating a robots localization state.

4.5 Result of Ada boosting

First we evaluate how well Ada Boosting using the features introduced in Section 3.3 performs on the classification task. The features represent information extracted from the particle filter and the actual perception. Using the discrimination metric 26 promising features out of 33 were selected for boosting. In this evaluation no output of the neural networks had been used.

In order to find the optimal number of weak classifiers learned in boosting we varied the maximum number between 26 and 300. Figure 8a shows the performance of the learned classification on the training and evaluation data

Table 5 Result of the mid-complex CNN

Environment	tp	tn	fp	fn	Accuracy	Precision	Recall
Training set	70 050	72 656	2 343	2 949	96.47%	96.76%	95.96%
Same environment	59 655	65 220	9 770	15 344	83.25%	85.92%	79.54%
New environment	48 284	51 113	23 886	26 715	66.27%	66.90%	64.38%

Fig. 7 The loss while training a mid-complex Convolutional Neural Network structure

Table 6 Separation results using a SVM (a) without the neural network feature (b) with the neural network feature

Feature	Set	tp	tn	fp	fn	Accuracy
(a) without NN feature	training set	23 027	23 099	14 291	14 572	61.51%
	same environment	23 158	23 289	14 320	14 222	61.94%
	new environment	21 735	21 970	15 639	15 864	58.11%
(b) with NN feature	training set	15 796	31 499	5 950	21 725	63.09%
	same environment	15 943	31 727	5 823	21 477	63.59%
	new environment	22 017	21 873	15 736	15 582	58.35%

Fig. 8 Training and test accuracy of AdaBoost with and without neural network feature on a different number of weak classifiers

Table 7 Boosting result using AdaBoost (a) without the neural network feature and 276 weak classifiers (b) with the neural network feature and 229 weak classifiers

Feature	Set	tp	tn	fp	fn	Accuracy
(a) without NN feature	training set	64 258	63 883	11 106	10 731	85.45%
	same environment	64 423	63 726	11 263	10 566	85.44%
	new environment	48 935	61 540	5 898	18 626	81.83%
(b) with NN feature	training set	33 633	32 563	4 886	3 888	88.30%
	same environment	33 675	32 453	5 097	3 745	88.21%
	new environment	51 550	59 988	7 450	12 011	82.62%

set. We selected 276 weak classifiers as an optimal number. Table 7a shows the performance of the so learned classifier on the training and the evaluation sets. The performance for the unknown environment is with 81.83% significantly better than the trained neural networks. But this comes at a price as the false negative rate is significantly higher than the false positive rate. This can be a problem for robots acting in an open environment.

Figure 8b shows the same learning and evaluation process where additionally the output of the mid-complex CNN was used as a feature. Here the optimal number of weak classifiers is 229 as the accuracy on the validation set does not significantly increase with further classifiers. Table 7b shows the performance of the so learned classifier on the training and the evaluation sets. It is clearly visible that the additional feature allows only a marginal performance improvement of about 1% on the data from the novel environment. However, the distribution of false negatives and false positives is now more balanced.

5 Conclusion and future work

Estimating its own position in an environment is a crucial capability of intelligent mobile robots. Knowing reliably the state of the localization process is important for the dependability of such robot systems.

Following the observation that most robots use distance sensors, environment maps, and particle filter based approaches we investigate if information generated from these components (patterns and statistical features of the particle set) can be used to train a reliable localization monitor. Using training data from test runs in a simulation environment where delocalization was provoked we trained several deep networks (non-recurrent and recurrent) and a boosting approach. While the trained deep networks showed moderate classification rates in particular in an environment not used for training and the SVM delivers insufficient results, the adaptive boosting approach using the network output as an additional feature showed detection rates of more than 80%. In conclusion, the evaluation of different training settings showed that it is possible to use information obtained from the particle set for scoring the localization accuracy.

For future work it needs to be investigated if delocalization caused by other uncertainties such as slipping can be detected as well. Moreover, a more detailed investigation of networks structures needs to be done. As the results show a significant decrease in accuracy when using the approach in a different environment, an investigation is to be conducted to see whether the approach can be generalized. Finally, so far only the position of particles is used for training.

It would be interesting if their orientation bears additional useful information.

Funding Open Access funding provided by Graz University of Technology. This research was done in cooperation with the company IncubedIT GmbH, which provided the hardware for the data collection.

Availability of data and material The data supporting the results of this study are available in the paper. Additional data generated during the project can be obtained from the corresponding author after consultation with the supporting company.

Code availability The data supporting the results of this study are available in the paper. Additional data generated during the project can be obtained from the corresponding author after consultation with the supporting company.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Thrun S, Burgard W, Fox D (2005) Probabilistic robotics (intelligent robotics and autonomous agents). The MIT Press
2. Thrun S, Fox D, Burgard W, Dellaert F (2001) Robust monte carlo localization for mobile robots. *Artif Intell* 128(1):99–141
3. Rubinstein RY, Kroese DP (2007) Simulation and the monte carlo method (wiley series in probability and statistics), 2nd edn. Wiley
4. Robert CP, Casella G (2005) Monte carlo statistical methods (springer texts in statistics). Springer, New York
5. Metropolis N, Ulam SM (September 1949) The monte carlo method. *J Am Stat Assoc* 44(247):335–341
6. Durrant-Whyte H, Bailey T (2006) Simultaneous localization and mapping: part i. *IEEE Robot Autom Mag* 13(2):99–110. <https://doi.org/10.1109/MRA.2006.1638022>
7. Doucet A, Johansen AM (2011) A tutorial on particle filtering and smoothing: fifteen years later
8. Turner L, Sherlock C (2013) An introduction to particle filtering. Technical Report, pp 1–25
9. Salimpour Y, Soltanian-Zadeh H (2009) Particle filtering of point processes observation with application on the modeling of visual cortex neural spiking activity. In: 2009 4th International IEEE/EMBS Conference on Neural Engineering, pp 718–721
10. Djuric PM, Bugallo MF (2009) Estimation of stochastic rate constants and tracking of species in biochemical networks with second-order reactions. In: 2009 17th European Signal Processing Conference, pp 2308–2311
11. Arulampalam MS, Maskell S, Gordon N, Clapp T (2002) A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans Signal Process* 50(2):174–188. <https://doi.org/10.1109/78.978374>

12. Kim S, Shephard N, Chib S (1998) Stochastic volatility: Likelihood inference and comparison with arch models. *Rev Econ Stud* 65(3):361–393
13. de J. Mateo Sanguino T, Gomez FP (2016) Toward simple strategy for optimal tracking and localization of robots with adaptive particle filtering. *IEEE/ASME Trans Mechatron* 21(6):2793–2804
14. Gu S, Ghahramani Z, Turner RE (2015) Neural adaptive sequential monte carlo. *CoRR arXiv:1506.03338*
15. Fox D (2002) Kld-sampling: Adaptive particle filters. In: Dietterich TG, Becker S, Ghahramani Z (eds) *Advances in Neural Information Processing Systems*, vol 14. MIT Press, pp 713–720
16. Röwekämper J, Sprunk C, Tipaldi GD, Stachniss C, Pfaff P, Burgard W (2012) On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 3158–3164
17. Müller M, Röder T, Clausen M, Eberhardt B, Krüger B, Weber A (2007) Documentation mocap database hdm05. Technical Report CG-2007-2. Universität Bonn
18. Wang X, Jia Y, Xi N, Zhen J, Xu F (2013) Mobile robot pose estimation using laser scan matching based on fourier transform. In: 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp 474–479
19. Kallasi F, Rizzini DL, Caselli S (2016) Fast keypoint features from laser scanner for robot localization and mapping. *IEEE Robot Autom Lett* 1(1):176–183. <https://doi.org/10.1109/LRA.2016.2517210>
20. Tipaldi GD, Braun M, Arras KO (2014) Flirt: Interest regions for 2d range data with applications to robot navigation. In: *Experimental Robotics: The 12th International Symposium on Experimental Robotics*. Springer, Berlin, pp 695–710
21. Oliveira GL, Radwan N, Burgard W, Brox T (2020) Topometric localization with deep learning. In: Amato NM, Hager G, Thomas S, Torres-Torriti M (eds) *Robotics Research*. Springer International Publishing, Cham, pp 505–520
22. Havangi R (2019) Mobile robot localization based on pso estimator. *Asian J Control* 21(4):2167–2178. <https://doi.org/10.1002/asjc.2004>
23. Güler S, Abdelkader M, Shamma JS (2019) Infrastructure-free multi-robot localization with ultrawideband sensors. In: 2019 American Control Conference (ACC), pp 13–18
24. Arana GD, Hafez OA, Joerger M, Spenko M (2019) Recursive integrity monitoring for mobile robot localization safety. In: 2019 International Conference on Robotics and Automation (ICRA), pp 305–311
25. Hafez OA, Arana GD, Joerger M, Spenko M (2020) Quantifying robot localization safety: A new integrity monitoring method for fixed-lag smoothing. *IEEE Robot Autom Lett* 5(2):3182–3189
26. Shamov IA, Shelest PS (2017) Application of the convolutional neural network to design an algorithm for recognition of tower lighthouses. In: 2017 24th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), pp 1–2
27. Piewak F, Rehfeld T, Weber M, Zöllner JM (2017) Fully convolutional neural networks for dynamic object detection in grid maps. In: 2017 IEEE Intelligent Vehicles Symposium (IV), pp 392–398
28. Hochreiter S, Schmidhuber J (November 1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
29. Gensler A, Henze J, Sick B, Raabe N (2016) Deep learning for solar power forecasting. an approach using autoencoder and lstm neural networks. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp 002858–002865
30. Carata SV, Neagoe VE (2016) A pulse-coupled neural network approach for image segmentation and its pattern recognition application. In: 2016 International Conference on Communications (COMM), pp 61–64
31. Dolezel P, Skrabanek P, Gago L (2016) Pattern recognition neural network as a tool for pest birds detection. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp 1–6
32. Morales N, Toledo J, Acosta L, Sánchez-Medina J (2017) A combined voxel and particle filter-based approach for fast obstacle detection and tracking in automotive applications. *IEEE Trans Intell Transp Syst* 18(7):1824–1834
33. Wang J, Chen X, Lu H, Zhang L, Pan J, Bao Y, Su J, Qian D (2020) Feature-shared adaptive-boost deep learning for invasive-ness classification of pulmonary subsolid nodules in ct images. *Med Phys* 47(4):1738–1749. <https://doi.org/10.1002/mp.14068>
34. Kotu V, Deshpande B (2019) Chapter 8 - model evaluation. In: Kotu V, Deshpande B (eds) *Data Science*. 2nd edn. Morgan Kaufmann, pp 263–279. <http://www.sciencedirect.com/science/article/pii/B9780128147610000083>
35. Duda RO, Hart PE (January 1972) Use of the hough transformation to detect lines and curves in pictures. *Commun ACM* 15(1):11–15
36. Eder M (2017) Using particle filters and machine learning approaches for state estimation on robot localization scoring. Master's Thesis, Faculty for Computer Science and Biomedical Engineering, Graz University of Technology
37. Polani D (2013) Kullback-leibler divergence. In: *Encyclopedia of Systems Biology*. Springer, New York, pp 1087–1088
38. Schapire RE (1999) A brief introduction to boosting. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Vol 2, IJCAI'99. Morgan Kaufmann Publishers Inc., San Francisco, pp 1401–1406
39. Schapire RE (2013) Explaining adaboost. In: *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*. Springer Berlin Heidelberg, Berlin, pp 37–52
40. Friedman J, Hastie T, Tibshirani R (2000) Additive Logistic Regression: a Statistical View of Boosting. In: *Ann Stat* 38(2)
41. Nock R, Nielsen F (2007) A real generalization of discrete adaboost. *Artif Intell* 171(1):25–41
42. Steinwart I, Christmann A (2008) Support vector machines, 1st edn. Springer Publishing Company, Incorporated
43. Hsu C-W, Chang C-C, Lin C-J (2003) A practical guide to support vector classification. Technical Report, Department of Computer Science, National Taiwan University. <http://www.csie.ntu.edu.tw/~cjlin/papers.html>
44. Cover TM (1965) Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans Electron Comput* EC-14(3):326–334. <https://doi.org/10.1109/PGEC.1965.264137>
45. Gerkey B, Vaughan RT, Howard A (2003) The player/stage project: Tools for multi-robot and distributed sensor systems. In: *Proceedings of the 11th International Conference on Advanced Robotics*, pp 317–323
46. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) Ros: an open-source robot operating system. In: *ICRA Workshop on Open Source Software*

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Matthias Eder completed his master's degree in Software Engineering and Management at Graz University of Technology. The focus of his studies was the conceptual design and development of intelligent systems. Currently, he is a scientific assistant at the Institute of Software Technology at Graz University of Technology. He works in the research group for autonomous intelligent systems, where he is also doing his PhD. His research focus in mobile robotics is on

explainable artificial intelligence to enhance the transparency and trustworthiness of intelligent systems.

Gerald Steinbauer is Associate Professor at the Institute for Software Technology at Graz University of Technology and lead the research group on Autonomous Intelligent Systems at the institute. His research interest is the application of AI techniques to robotics in particular to improve the dependability of mobile robot systems. Application areas range from transport robots over flexible production to field robotics. He is an active member of the

RoboCup community, founded several RoboCup teams and serves currently as a member of the board of the RoboCup Federation.

Michael Reip did his PhD at Graz University of Technology working on Dependable Belief Management for intelligent mobile robots. During his studies he was active in RoboCup, letting robots play soccer. Together with his colleagues from the RoboCup team, he founded the company incubed IT in 2011. Using the expertise in mobile robotics, the team could grow the company to provide leading software for mobile robots in the industry. He developed soft-

ware, directed the development direction and, defined the company strategy.