Deep Learning-Based Mutual Detection and Collaborative Localization for Mobile Robot Fleets Using Solely 2D LIDAR Sensors

Robin Dietrich[1] and Stefan Dörr[1]

Abstract— **Localization for mobile robots in dynamic, large-scale environments is a challenging task, especially when relying solely on odometry and 2D LIDAR data. When operating in fleets, mutual detection and the exchange of localization information can be highly valuable. Detecting and classifying different robot types in a heterogeneous fleet, however, is non-trivial with 2D LIDAR data due to the sparse observation information.**

In this paper a novel approach for mutual robot detection, classification and relative pose estimation based on a combination of convolutional and ConvLSTM layers is presented in order to solve this issue. The algorithm learns an end-to-end classification and pose estimation of robot shapes using 2D LIDAR information transformed into a grid-map. Subsequently a mixture model representing the probability distribution of the pose measurement for each robot type is extracted out of the heatmap output of the network. The output is then used in a cloud-based collaborative localization system in order to increase the localization of the individual robots.

The effectiveness of our approach is demonstrated in both, simulation and real-world experiments. The results of our evaluation show that the classification network is able to achieve a precision of 90% on real-world data with an average position estimation error of 14 cm. Moreover, the collaborative localization system is able to increase the localization accuracy of a robot equipped with low-cost sensors by 63%.

I. INTRODUCTION

Mobile robots are a key component to realize a flexible, efficient, and scalable material flow in logistics and production sites. Due to the individual requirements of different transport tasks, we commonly find heterogeneous fleets sharing workspaces with humans and human controlled vehicles, like forklifts. For navigation, 2D LIDAR sensors are still predominant since they have proven suitable robustness and reliability for these kinds of environments and even more important, simultaneously provide safe collision avoidance with humans. Moreover, the design of the robots to carry loads often forbids mounting further sensors like camera systems at suitable positions.

However, localizing the robot with 2D LIDAR data is challenging in these dynamic and large-scale environments, especially when dealing with noisy sensor data from safety or low-cost scanners. In our previous work [1], we tackled this issue with a cloud-based cooperative long-term Simultaneous Localization and Mapping (SLAM) approach where each robot as well as further available static sensors transmit detected changes of the environment to a map server which fuses the information into a consistent global map and provides map updates for the mobile robots. These map updates have been shown to be crucial to maintain an accurate and robust localization of the robot, also in highly changing environments. However, as for map-based localization approaches in general, it requires a sufficient density of static and semi-static objects within the environment visible for the sensors that are typically mounted at leg height. In areas with sparse geometrical structure, an up-to-date map cannot help much to decrease localization uncertainty. This circumstance gets worse when using low-cost sensors with a limited field of view.

In this work, we approach this issue by adding a mutual localization capability to our cooperative system exploiting the property that robots operating in fleets and shared workspaces constantly observe each other. Extracting other robots from sensor observations and processing the relative pose measurements can thereby overcome the discussed problems in sparse structured areas. In contrast to similar work [2][3][4][5][6][7], we follow a solely 2D LIDAR- and non-marker-based approach.

We leverage the recent advances in deep learning for object detection and classification [8][9] as well as their demonstrated ability of tracking objects in image data [10][11] using fully-convolutional networks. We combine this concept with recurrent layers - the ConvLSTM [12] - in order to maintain the dynamic state of the environment and distinguish moving from none-moving objects over time. To the best of our knowledge there has only been one similar approach for the detection and classification of objects in a 2D LIDAR scan using a deep learning approach by Ondruska et al. [13][14][15]. These algorithms, however, assume either a static sensor or do not classify the objects at all, rendering it incompatible for our demands. Using a deep learning approach allows us to improve the processing of noisy sensor data, decreases manual configuration and parametrization efforts as well as deal with arbitrary, a priori unknown robot shapes. As an example, slight differences in the shape of a robot over time, caused by e.g. height differences in the source scanner, can be compensated by gathering enough training data instead of specifying multiple different shapes for a scan matching.

Consequently, we demonstrate how the mutual detections are used to improve the localization estimates of the respective robots. The main contribution of the paper is:

- A deep-learning based detection module using both, convolutional and recurrent layers for a detection, classification and pose estimation of different robot types.

[1]Fraunhofer Institute for Manufacturing Engineering and Automation IPA Stuttgart, Robin.Dietrich, Stefan.Doerr @ipa.fraunhofer.de

- A probabilistic detection model incorporating uncertainty of the relative pose measurements as well as the current localization estimate into a combined mixture model for an accurate and though efficient communication of relevant information through the network.
- A Monte-Carlo Localization (MCL)-based update of the pose beliefs of measured as well as detector robot handling unknown associations.

II. RELATED WORK

Many of the existing multi-robot localization approaches which incorporate relative pose estimates of other robots either assume a given detection module [4][16][6] or use straightforward methods like marker- or vision-based detection [2][7]. Approaches for classifying objects with a 2D LIDAR scanner can commonly be distinguished into model-based and learning-based approaches. Specifying a model for e.g., a subsequent scan matching for classification using the iterative closest point (ICP) is, however, a non-trivial task due to sensor noise and dependent on the number of classes [17]. Traditional learning-based approaches like Support Vector Machines (SVMs) remove the dependency on a specific model but instead rely on the definition of specific features [18].

In contrast, current learning-based approaches including Convolutional Neural Networks (CNNs) are able to learn suitable features for object detection and classification. After achieving outstanding scores on image classification [19][20][21][22] for years, the research focus in this field shifted towards end-to-end object detection, classification and semantic segmentation of images. By using fully-convolutional networks without any fully-connected layers, networks as presented by Long et al. [8] or Badrinarayanan et al. [23] are able to segment a complete image into different classes, one per pixel.

In the majority of the cases, these architectures have been applied to image or 3D pointcloud processing. Although the data processing for these sensors is computationally much more demanding, the task itself is less complex than it is using 2D LIDAR data, since 2D LIDARs only provide sparse measurement information compared to a 3D LIDAR or a camera. This makes it difficult to classify and track objects relying solely on this sensor.

Ondruska et al. [13][14][15] use similar techniques for the detection, classification and tracking of moving objects in a 2D occupancy grid-map generated from a 2D laser scanner. Their first approach [13] is able to detect and track moving objects even in occluded space using a Recurrent Neural Network (RNN) with convolutional operations. However, the source scanner is static, the field of view relatively small (50×50 pixels) and the objects are moving with a constant velocity. These constraints render the approach impractical for real-world applications. The second version of this approach is tested in a real-world scenario including an intersection [14]. While the source scanner is still static, the objects are now diverse in shape and movement (pedestrians, bicycles, cars). By using a convolutional variant of the Gated Recurrent Unit (GRU) including a bias per neuron, the network is able to detect and classify objects. This works only in the learned environment, since the bias is a static memory which learns the position of an object class in the image rather than detecting it by its shape. The latest improvement of this system is presented by Dequaire et al. [15], who apply it to a moving vehicle. The authors add a spatial transformer module to the previous network to account for the motion of the ego vehicle. The network still detects and tracks objects but it is no longer capable of classifying them, since the static memory approach does not work in this case. To the best of our knowledge, there does not exist any prior approach for object detection, classification and pose estimation from a moving vehicle using only 2D LIDAR data with convolutional RNNs.

For incorporating the relative pose measurements into a collaborative localization system, the approaches from Fox et al. [2] and Prorok et al. [24] are most relevant for this work. Both build upon the well-known MCL as the core algorithm and expand it to be able to process mutual detections. The approaches mainly differ in the way the particle set is transformed and communicated through the network. While Fox et al. [2] use density trees, Prorok et al. [24] propose a cluster-based approach. Both, however, do not address the problem of data association, i.e. they assume the identity of the detected robot to be known which can rarely be met when basing on 2D LIDAR sensors.

III. MUTUAL DETECTION

Interpreting the raw data of a 2D LIDAR is a challenging task for humans as well as machines. We therefore transform the output of the sensor into a 2D occupancy grid-map. For simplicity we only label occupied (1) and free (0) cells. Traditionally unknown space is labeled as well in an occupancy grid-map, but experiments with and without unknown space have shown that the unknown space does not increase the performance of the system significantly while requiring an expensive ray-tracing for each occupied cell [25]. The unknown space is therefore neglected in this work.

A. Detection and Classification

In order to estimate a position for each robot, as well as classify its type, a fully-convolutional network [26] is introduced. This allows the network to maintain spatial information from the input and project it to the output. This property is crucial for enabling an accurate position estimation. The dimensionality of the input is $n \times n \times 1$, where n is the size of the map, calculated from the laserscan in both x, and y dimension. The detector robot R_d is located at the center of the map to allow a full 360-degree representation of the robot's surroundings, independent from the robot's angular sensor range. The output $n \times n \times (c+1)$ maintains the height and width (n) of the map while increasing the *frame*-dimension depending on the number of classes (c). The output therefore maintains a map for each robot type as well as one for the *rest* class (everything else). The position of each robot in the output is marked by a filled circle around

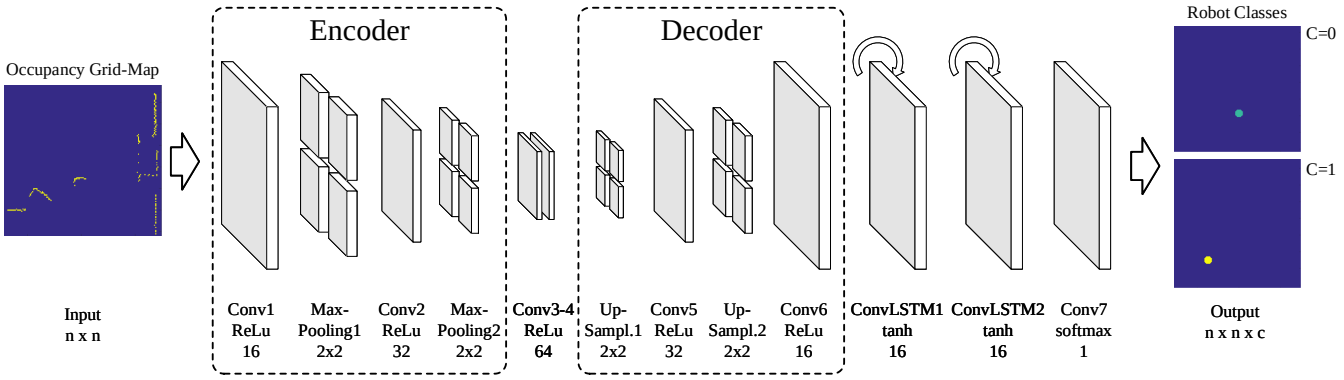


Fig. 1: The architecture and in-/output of the developed CNN/ConvLSTM network featuring end-to-end object classification and position estimation.

the actual robot position, large enough to overlap with the robots' scan points. An example for an input and output of the network for two robot types is shown in Fig. 1, together with the network's architecture.

Detecting a robot with a e.g. rectangular shape and distinguishing it from a static box is hard while it is not moving. But since the pose estimate of each robot is only updated when it is moving, we do not need to detect non-moving robots. Therefore, we use ConvLSTMs, which maintain information in the network over time, in order to track moving robots. ConvLSTMs are able to represent the dynamic state of an object internally [14] and are therefore suitable for detecting dynamic objects. We use *stateful* LSTMs, i.e. LSTMs where the state at time t depends on the previous state $t-1$. This allows us to train the network using consecutive scans without resetting the internal state of the network after each sample. The network is thus able to increase the detection and classification performance, since a robot generally stays within its receptive field in two consecutive scans due to its low velocity $(1-2m/s)$ in combination with the sensors frequency $(10Hz)$. It further enables the network to process data from a moving ego-vehicle without processing its velocity information explicitly because it is able to maintain an odometry-like estimate for the ego-vehicle internally. This is demonstrated by the experiments conducted in section V-B.

The first part of the network is an encoder-decoder architecture similar to the one presented by Badrinarayanan et al. [23] for end-to-end pixel-wise classification of images. These layers are responsible for learning features for the classification of robots. The following ConvLSTMs then process these features in order to maintain the internal state of the robot and track its position over time. A convolutional layer with a softmax activation finally outputs the position probability for each robot hypothesis.

B. Pose Estimation

As shown in the previous part of this work, the output of the network can be viewed as a map with multi-modal gaussians at the detected robots' mean position estimate. As a straightforward approach for estimating multiple position hypotheses for each robot class, we therefore use a standard hierarchical clustering algorithm. The two-dimensional output array for each class is clustered into r different clusters, where only those cells exceeding a defined threshold s are considered. Afterwards, each cluster is processed separately and a mean position as well as a probability p_d for the cluster is generated based on the number of points detected for this hypothesis, and the networks estimated probability for each of these points.

A second network, trained on the output of the classification network, is used in order to estimate an orientation for each cluster in each robots' output. For each hypothesis in a robots' output, a map around the estimated position is extracted and used as an input to the orientation network. The architecture is composed of three convolutional layers, each followed by a max-pooling operation in order to reduce the size of the data from $q \times q \times 1$ to $a \times 1 \times 1$, where q denotes the size of the map part around the robot and a denotes the number of possible angle values. We use a discretized representation of the orientation, since common robot shapes (rectangular/circular) appear ambiguous in the scan due to their manifold rotation symmetry. The network therefore outputs a probability distribution of the robot's orientation over a angles. For a predefined number of best angular estimates a_{best} a Gaussian Mixture Model (GMM) is created with component probability w_i depending on the networks probability output for the respective angle.

C. Measurement Representation

With the combined position and orientation estimation, the detection module is able to output the relative pose measurement $z_t^{(m,d)}$ of measured robot R_m relative to detector robot R_d at time t. For representing the uncertainty of the measurement, we choose the following mixture model with model parameters θ_t^z:

$$p(z_t^{(m,d)} \mid \theta_t^z) = p_d \sum_{i=1}^{N} w_i \; p(z_t^{(m,d)} \mid \mu_i, \Sigma_i) + p_{fp}. \quad (1)$$

Herein, p_d is the detection confidence describing the probability that we actually detected a robot of type R_m.

Additionally, the probability of a false-positive measurement p_{fp} is incorporated. This parameter is constant and extracted from the learning phase of the network in the specific environment. The core of Eq. (1) is then a GMM which models the spatial uncertainty of the measurement with mean μ, covariance Σ and weight w of each component of the GMM and by using the Gaussian probability density function (PDF) in $p(z_t^{(m,d)} \mid \mu_i, \Sigma_i)$. The advantage of using a mixture model compared to non-parametric approaches (like e.g. density trees) lies in the low amount of data needed to describe the measurement distribution while still being able to approximate relevant distributions with an appropriate accuracy. This is particularly significant since we need to communicate the measurement data over the wireless network as described in the following section.

IV. CLOUD-BASED COLLABORATIVE LOCALIZATION

This section presents the integration of the relative pose measurements into a collaborative localization system in order to improve the localization of the robots. The approach builds upon our cloud-based multi-robot navigation system presented in [27] where each robot is connected to the navigation server enabling the possibility to share knowledge among the fleet as well as to exploit the massive computational resources in the cloud for remote computations.

From the localization perspective, each robot runs its own long-term SLAM module providing localization information for its local navigation system based on its on board 2D Lidar and odometry data. The robots further share detected map changes with the map server in the cloud. The map server fuses the incoming map information into a consistent global map which is then distributed to the robots in order to increase the robustness and precision of the long-term SLAM in changing environments, see [1] for detailed information.

Within this work and in contrast to [1], an approach similar to [28] is used for long-term SLAM where MCL on a dual layered map serves as the core localization algorithm. As in MCL, the belief about the 2D pose x_t (position + orientation) of the robot is approximated with a set of particles:

$$bel(x_t) \approx \sum_{k=1}^{N_k} w_t^{[k]} \delta \left(x_t - x_t^{[k]} \right) \tag{2}$$

where δ is the Dirac function, $x_t^{[k]}$ the particle's sample of the state space and $w_t^{[k]}$ its (importance) weight. To update the pose belief at time t, the particles are sampled from the motion model based on current odometry information and subsequently weighted based on the current sensor observation z_t and the observation model:

$$w_t^{[k]} = p(z_t \mid x_t^{[k]}). \tag{3}$$

Following [2], the relative pose measurement $z_t^{(m,d)}$ can be can incorporated into the pose belief of R_m with:

$$bel(x_t^{(m)}) \rightarrow bel(x_t^{(m)}) \int p(x_t^{(m)} \mid z_t^{(m,d)}, x_t^{(d)})\, bel(x_t^{(d)})\, dx_t^{(d)}. \tag{4}$$

Due to its symmetry, Eq. (4) can also be used to update the pose belief of R_d just by switching respective terms. The detection model $p(x_t^{(m)} \mid z_t^{(m,n)}, x_t^{(m)})$ can be derived by transforming the relative pose measurement model given in Eq. (1) into the global frame using the pose $x_t^{(d)}$ of R_d:

$$p(x_t^{(m)} \mid z_t^{(m,d)}, x_t^{(d)}) = p(z_t^{(m,d)} \mid \hat{\theta}_t) \tag{5}$$

where $\hat{\theta}$ are the transformed model parameters, i.e. $\hat{\mu_i} = T(x_t^{(d)})\mu_i$ and $\hat{\Sigma}_i = T(x_t^{(d)})\Sigma_i T(x_t^{(d)})^T$, given the transformation matrix $T(x_t^{(d)})$ based on the pose of R_d.

For realizing the pose belief update of Eq. (4), R_m needs knowledge about θ_t and $bel(x_t(d))$ (i.e. the particle set). However, communicating the particle set through the wireless network is inefficient, especially when using large particle numbers typically needed when dealing with highly noisy sensor data. Moreover, it would imply multiplicating two particle sets in Eq. (4) which cannot be done straightforwardly. Instead, we approximate $bel(x_t^{(d)})$ with a more memory-efficient representation in terms of a GMM:

$$bel(x_t^{(d)}) \approx p(x_t \mid \theta_t^{(d)}) = \sum_{j=1}^{N_j} w_j\, p(x_t^{(d)} \mid \mu_j, \Sigma_j). \tag{6}$$

The computation of the GMM parameters $\theta_t^{(d)})$ from the particle set is carried out using an Expectation Maximization (EM) algorithm as in [29]. In our experiments, we found that in most situations the particle set can be well approximated with a GMM containing only few components supporting the usage of the GMM as both, an accurate and compact representation for this particular purpose.

With Eq. (5) and (6), we can now compute the integral in Eq. (4) according to:

$$\int p(x_t^{(m)} \mid z_t^{(m,d)}, x_t^{(d)})\, bel(x_t^{(d)})\, dx_t^{(d)} = p_d \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} w_{i,j}\, p(x_t^{(m)} \mid \mu_{i,j}, \Sigma_{i,j}) + p_{fp} \tag{7}$$

with $w_{i,j} = w_i w_j$, $\mu_{i,j} = T(\mu_j)\mu_i + \mu_j$ and $\Sigma_{i,j} = T(\mu_j)\Sigma_i T(\mu_j)^T + \Sigma_j$. In the following, we use Θ_t to describe the resulting parameter set from Eq. (7).

Updating the particles of R_m then simplifies to inserting the respective particle pose $x_t^{[k](m)}$ into Eq. (7) to compute its weight:

$$w^{[k](m)} = p(x_t^{[k](m)} \mid \Theta_t) = p_d \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} w_{i,j}\, p(x_t^{[k](m)} \mid \mu_{i,j}, \Sigma_{i,j}) + p_{fp}. \tag{8}$$

Reversely, for updating the pose belief of R_d, R_m only needs to communicate back its current pose belief in GMM

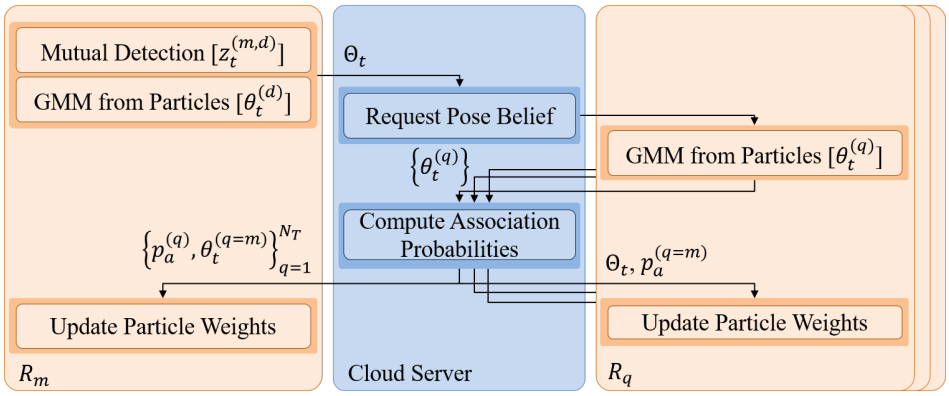


Fig. 2: Schematic illustration of collaborative localization with unknown associations.

representation. R_d can then also apply Eq. (8) to its particle set.

So far, we assumed that we have knowledge about the identity of R_m. However, clearly identifying the robots of the fleet based on 2D LIDAR data would require each robot having an unambiguous, clearly distinguishable shape with respect to the other robots. This is never the case in homogeneous fleets. Moreover, also in heterogeneous fleets, most robots commonly appear similar due to the currently predominate rectangular or circular shape of most mobile robots. To resolve this issue, we need to deal with data association, i.e. finding the robot in the fleet which was detected. In our cloud-based architecture, this task is carried out on the cloud server. Once a detection is received from R_d, the cloud server requests the GMM-based pose estimates of the subset of robots T with the particular type that was detected. It then computes the association probability $p_a^{(q=m)}$ of robot R_q being the one that was measured by R_d:

$$\begin{aligned} p_a^{(q=m)} &= p(\theta_t^{(q)} \mid \Theta_t) \\ &= \eta \sum_{j=1}^{N_j} \sum_{i,j=1}^{N_i,N_j} w_j^{(q)} w_{i,j}^{(d)} p(\mu_i^{(q)} \mid \mu_{i,j}^{(d)}, \Sigma_{i,j}^{(d)} + \Sigma_i^{(q)}). \end{aligned} \quad (9)$$

where η is the normalizer:

$$\eta^{-1} = \left(\sum_{v=1}^{N_T} p_a^{(v=m)} \right) + p_{fp}. \quad (10)$$

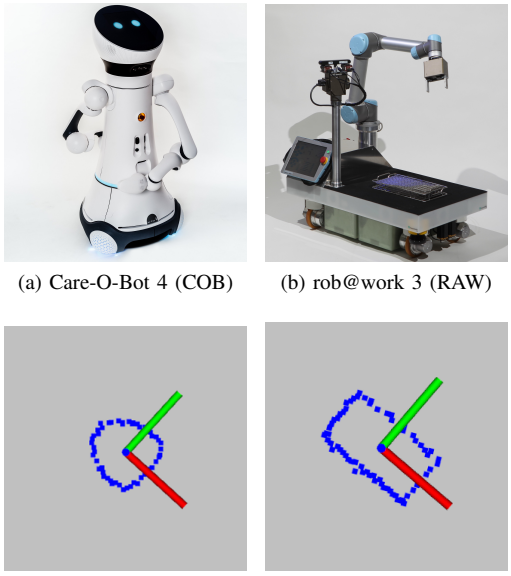
Each robot then updates its particles weights based on its association probability:

$$w^{[k](q)} = p_a^{(q=m)}\ p(x_t^{[k](q)} \mid \Theta_t) + (1 - p_a^{(q=m)}). \quad (11)$$

The concept for mutual localization with unknown associations including the information flow from the robots to the cloud server and back is illustrated in Fig. 2.

V. EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of the introduced networks and algorithms, we will demonstrate their performance in different environments. Although a comparison to a classic approach like scan matching would be highly desirable, this is not a straightforward task. Unlike e.g. for the robot's localization, there does not exist a standard package for detection, classification and position estimation available in the robot operating system (ROS) or other open source frameworks. Implementing and especially parameterizing an algorithm for this specific task in order to facilitate a real benchmark is a challenging and time-consuming task. We therefore decided to do an in-depth evaluation of the detection, classification and position estimation network instead, which is partially included in this section and fully available in [25]. The thesis includes without limitation an evaluation of the network architecture (FCN vs. ConvLSTM), the impact of localization uncertainty and a detailed comparison between simulated and real-world data.



(a) Care-O-Bot 4 (COB) (b) rob@work 3 (RAW)

(c) COB 2D LIDAR shape (d) RAW 2D LIDAR shape

Fig. 3: The two robots used for the experiments (a, b), together with their respective shapes (c, d) as seen in a 2D LIDAR scan.

A. Experimental Setup

The approaches presented in the previous sections are evaluated using the real-world mobile robots, Care-O-bot 4[1] (COB, Fig. 3a/3c) and rob@work 3[2] (RAW, Fig. 3b/3d). Both are equipped with Sick S300 Professional safety 2D LIDARs for a full 360-degree view around the robot. Apart from testing on real hardware, we can rely on a realistic robot simulation using Gazebo[3]. While in simulation, ground truth data is available to measure the performance of the approaches, a map-based localization system given ground truth map data is used as the reference system in our real-world experiments.

B. Mutual Detection, Classification & Position Estimation

For the evaluation of the detection, classification and relative pose estimation module, two experiments will be presented, demonstrating the algorithms ability of generalizing to different simulation environments as well as its performance on real-world data. In both cases, the respective model is evaluated on four different environments: a simulated room known from the training data (A), an unknown simulated room (B), a known real-world environment (C) and an unknown real-world environment (D). The simulation rooms are a machine-hall (A) and a warehouse environment (B), the real-world environments are a large-scale industrial hall (C) and a smaller lab (D). For both experiments, two RAWs and a COB are used, where only the sensor data of one RAW is evaluated. All robots, including the ego-vehicle, are moving continuously during the experiments. The map size for the networks input is chosen to be 224×224 cells/pixels with a resolution of $0.05m$, resulting in a detection range of around $5.6m$.

The models used for evaluating the mutual detection, classification and position estimation have around 125000 parameters. The data-sets used for the supervised training of these networks consist of 21000 and 3000 examples for simulation and real-world, respectively. They have been gathered by autonomous runs through different environments. The data was split into a train (70%) and test set (30%) used to train the network with an Adam optimizer and evaluate its performance. For the training of the network, a sequence length of one was chosen in order to allow the online use of the network at each time step of a real-time run. This is possible, since the ConvLSTM was chosen to be stateful, meaning that the state at time t depends on the output of the network at $t-1$. Additional data-sets for the evaluation presented in this section were gathered with 3000 examples for each environment. The data used for evaluating the networks is therefore completely different from the training data, although collected in the same environments. The data gathered in the real-world for both, training and evaluation, includes people walking around occasionally, which has not influenced the performance of the networks in any noticeable way.

[1]https://www.care-o-bot.de/de/care-o-bot-4.html
[2]https://www.care-o-bot.de/de/rob-work.html
[3]http://gazebosim.org/

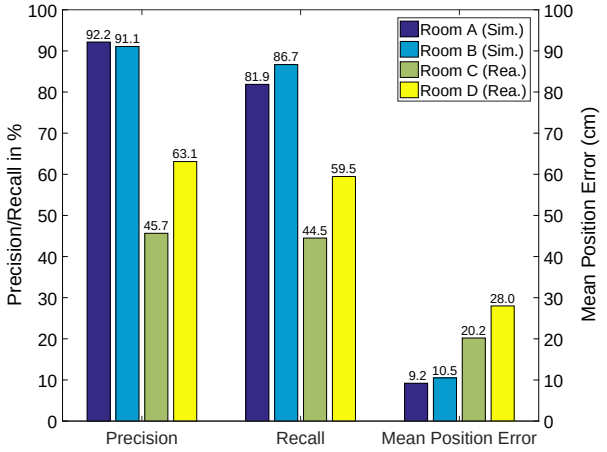


Fig. 4: The evaluation results of a classifier trained solely on simulation data.

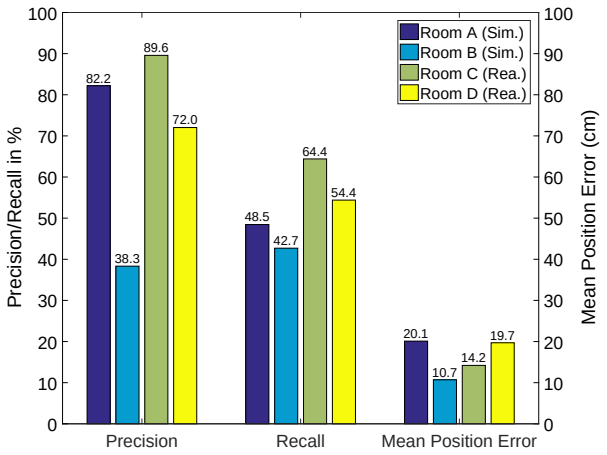


Fig. 5: The evaluation results of a classifier pre-trained on simulation data and continuously trained on real-world data.

The first test is conducted using a model trained on data gathered from multiple different simulated environments (including room A). The resulting data for this experiment is shown in Fig. 4. It demonstrates the model's ability of generalizing to different environments. When applied to an unknown environment the model is still able to classify more than 90% of the robots correctly (precision) while increasing the detection rate by ~5% (recall). When applied to real-world data, the overall performance decreases significantly, but stays at ~$50-60\%$. This shows the differences in the robots' shapes compared to simulated data. Generally, the shapes are very similar, since the simulation data is recorded using a 3D simulation of the robots' detailed CAD files. In real-world, however, sensors and enclosure parts may not be mounted 100% accurately. Additionally, small gradients or irregularities of the floor can lead to minor tilts of the robot and its sensor, which itself leads to a different shape

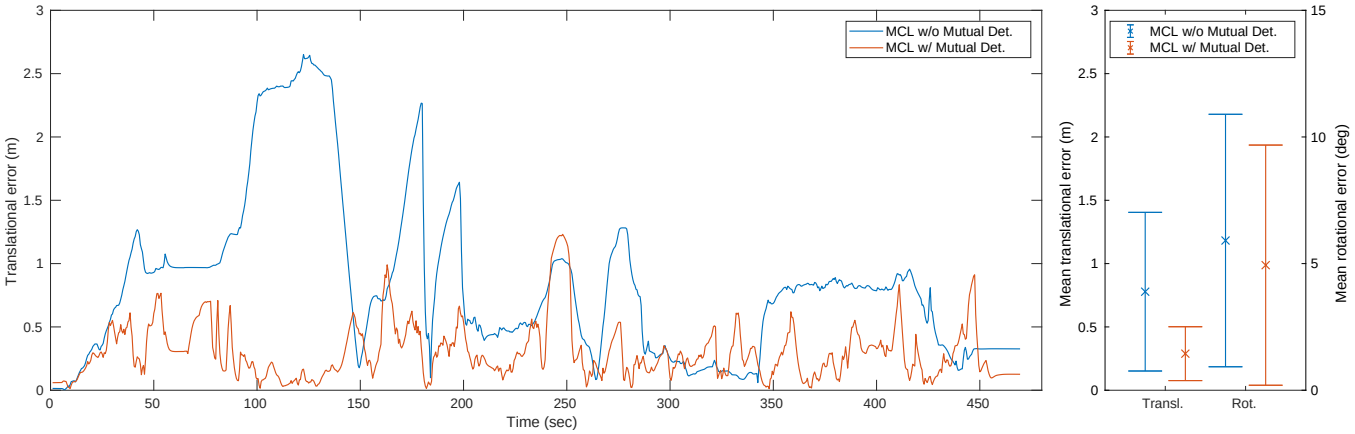


Fig. 6: The localization error using MCL with and without the mutual localization module. The left diagram shows the translational error over time while the right one displays the average translational and rotational error.

of the detected robot. Although this result is not yet sufficient for real-world scenarios, it demonstrates its potential to generalize in parts even to real-world applications. The mean position error of the relative pose measurement is ~$10cm$ for both simulated and ~$25cm$ for the real-world environment. While the real-world error is acceptable, the simulation error is comparable to map-based localization methods, especially when taking the robots' diameter (~$60-80cm$) into account.

Due to these limitations of the simulation model when applied to real-world data, we additionally trained the model on room C in order to adapt it to real-world data. The results for this experiment in Fig. 5 show that the performance on the known real-world room C increases significantly. The precision is now almost equally high as the precision of the previous model on known simulation data (room A). The performance on room D increases as well but there is still a big gap to the known data from room C. We theorize that the reason for is the insufficient amount of real-world data, since the model is only trained on one real-world environment compared to four different simulated rooms before. The performance on the simulated data decreases on the other hand since the model adapts to the real-world data.

For a more detailed evaluation of the robot detection, classification and position estimation module, we refer the reader to [25].

C. Mutual Localization

Our final experiment aims at evaluating the impact of the mutual detections on the localization accuracy. We investigate a scenario where a robot equipped with noisy, low-cost sensors operates in a sparsely structured environment together with multiple robots equipped with more sophisticated sensors. Since the performance of the detection and classification network in the real-world has already been proven in section V-B, this scenario is set up in simulation in order to have an accurate ground truth measure of the robots' poses. During the experiment, three COBs and one RAW navigate to random goals in a squared, empty room for about 450 seconds (~4500 data samples). In order to simulate the low-cost robot, we restrict the sensor range of the RAW's laser scanner to $5m$ and increase the simulated noise on the range measurements as well on its simulated odometry.

The results are visualized in Fig. 6 where the RAW's translational localization error is depicted for two configurations, with and without mutual localization (left), together with the average error in translation and rotation (right). Each robot starts in a corner of the room, therefore being well localized based on scan matching against the map. Once the RAW drives into the center of the room, after ~40 seconds, the localization without the mutual detection decreases significantly due to the accumulated uncertainty of the odometry data. The localization including detections from other robots, however, is significantly more stable. The difference is especially distinct in the time between 100 and 150 seconds, where the solely map-based localization fails due to the poor sensor setup of the robot. The mutual localization on the other hand is able to decrease the localization error to less than $5cm$. This behavior continues throughout the entire data-set, leading to a decrease in mean translational error by ~63% to ~$28cm$.

VI. CONCLUSION

In this paper, we presented a novel approach for mutual robot detection, classification, and relative pose estimation using a network with combined convolutional and ConvLSTM layers which outputs relative pose measurements in terms of probabilistic mixture models. These measurements were then integrated into an existing cloud-based collaborative long-term SLAM to enable mutual localization for a heterogeneous mobile robot fleet. In our experiments, we demonstrated that the algorithm is able to accurately detect and classify robots with different shapes while further estimating a relative pose of them in simulation as well as real-world scenarios. We further showed that the localization accuracy of a robot equipped with noisy sensors increased by 63% when operating in the same environment with multiple robots equipped with more sophisticated sensors.

Future work will deal with solving the 'kidnapped robot'

as well as global localization problem by using the mutual detections and sampling particles from its measurement model. Furthermore, we will investigate the behavior of the mutual localization system for larger fleet sizes and in real world scenarios.

References

[1] Stefan Dörr et al. *Cooperative Longterm SLAM for Navigating Mobile Robots in Industrial Applications*. In: *Proc. IEEE Int. Conf. Multisensor Fusion and Integration for Intelligent Systems (MFI)*. 2016.

[2] Dieter Fox et al. *A Probabilistic Approach to Collaborative Multi-Robot Localization*. In: *Autonomous Robots* 8.3 (2000).

[3] I. M. Rekleitis et al. *Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. Sept. 2002.

[4] S. I. Roumeliotis and G. A. Bekey. *Distributed multi-robot localization*. In: *IEEE Transactions on Robotics and Automation* 18.5 (Oct. 2002).

[5] Agostino Martinelli et al. *Multi-Robot Localization Using Relative Observations*. In: *IEEE International Conference on Robotics and Automation*. 2005.

[6] Antonio Franchi et al. *Mutual localization in a multi-robot system with anonymous relative position measures*. en. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. St. Louis, MO, USA: IEEE, Oct. 2009.

[7] Alicja Wasik et al. *Lidar-based relative position estimation and tracking for multi-robot systems*. In: *Robot 2015: Second Iberian Robotics Conference*. Springer. 2016.

[8] Jonathan Long et al. *Fully Convolutional Networks for Semantic Segmentation*. In: 2015.

[9] Pedro O. Pinheiro and Ronan Collobert. *From image-level to pixel-level labeling with convolutional networks*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[10] Qi Wang et al. *Generalized Path Corridor-based Local Path Planning for Nonholonomic Mobile Robot*. In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. Vol. 2015-Octob. Sept. 2015.

[11] Luca Bertinetto. *siamese-fc: Arbitrary object tracking at 50-100 FPS with Fully Convolutional Siamese networks*. original-date: 2016-08-30T16:13:13Z. Apr. 2018.

[12] Xingjian SHI et al. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015.

[13] Peter Ondruska and Ingmar Posner. *Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks*. In: *arXiv:1602.00991 [cs]* (Feb. 2016).

[14] Peter Ondruska et al. *End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Networks*. In: *arXiv:1604.05091 [cs]* (Apr. 2016).

[15] Julie Dequaire et al. *Deep Tracking on the Move: Learning to Track the World from a Moving Vehicle using Recurrent Neural Networks*. In: *arXiv:1609.09365 [cs]* (Sept. 2016).

[16] A. Martinelli et al. *Multi-Robot Localization Using Relative Observations*. en. In: IEEE, 2005.

[17] B. Douillard et al. *A Pipeline for the Segmentation and Classification of 3D Point Clouds*. en. In: *Experimental Robotics*. Ed. by Oussama Khatib et al. Vol. 79. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[18] B. Qin et al. *A Spatial-Temporal Approach for Moving Object Recognition with 2D LIDAR*. en. In: *Experimental Robotics*. Springer Tracts in Advanced Robotics. Springer, Cham, 2016.

[19] Kaiming He et al. *Deep Residual Learning for Image Recognition*. In: *arXiv:1512.03385 [cs]* (Dec. 2015). Microsoft ResNet.

[20] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. In: *arXiv:1409.1556 [cs]* (Sept. 2014).

[21] Christian Szegedy et al. *Going deeper with convolutions*. en. In: GoogLeNet. IEEE, June 2015.

[22] Alex Krizhevsky et al. *ImageNet Classification with Deep Convolutional Neural Networks*. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012.

[23] Vijay Badrinarayanan et al. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. In: *arXiv:1511.00561 [cs]* (Nov. 2015).

[24] Amanda Prorok et al. *Low-cost collaborative localization for large-scale multi-robot systems*. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, May 2012.

[25] Robin Dietrich. *Deep Learning Based Mutual Robot Detection and Relative Position Estimation*. English. Master-Thesis. University of Stuttgart, July 2018.

[26] E. Shelhamer et al. *Fully Convolutional Networks for Semantic Segmentation*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017). 01385.

[27] Jannik Abbenseth et al. *Cloud-based cooperative navigation for mobile service robots in dynamic industrial environments*. In: *Proceedings of the Symposium on Applied Computing - SAC '17*. ACM, 2017.

[28] Rafael Valencia et al. *Localization in highly dynamic environments using dual-timescale NDT-MCL*. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014.

[29] Israel Dejene Gebru et al. *EM Algorithms for Weighted-Data Clustering with Application to Audio-Visual Scene Analysis*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.12 (Dec. 2016).