Improving Obstacle Avoidance in End-to-End Deep Learning by Incremental Training on Collisions

Alexander K. Seewald[1] [0000−0002−9982−432X]

Seewald Solutions GmbH, Lärchenstraße 1, A-4616 Weißkirchen a.d. Traun, Austria
alex@seewald.at https://seewald.at

Abstract. Obstacle avoidance is an essential feature for autonomous robots. Although it is possible to train end-to-end deep learning models on this task, performance is not always competitive to specialized sensors, and the necessity for human-generated training data makes building such systems complex and costly. Here, we propose an incremental training method starting at the model described in [12]. By successively incrementally training the model on stereo images taken just before observed collisions, the collision rate can be significantly reduced after just a few iterations. Additionally, since training is stopped relatively early, computational effort is much lower than for a more traditional full training on the original and the new collision stereo images. No human-generated training data is needed and human intervention is minimal. A test with a model retrained on all data may even indicate that our method actually performs significantly better than full training.

Keywords: Deep Learning · Obstacle Avoidance · Autonomous Robotics · Mobile Robotics · End-to-End Learning

1 Introduction

Obstacle avoidance is an essential feature for autonomous robots, the lack of which can make the estimation of the robot's intelligence drop dramatically. As benchmark task it has been known from the beginning of autonomous robotics and is usually solved with specialized sensors and Simultaneous Localization and Mapping algorithms (SLAM, [3]). However, different sensors have different error modes and multiple sensor classes must be combined to obtain acceptable results; and SLAM algorithms rely on complex map-building, are not robust enough for a wide variety of environments, and are very tricky to set up, adjust and make sufficiently robust for even one given environment.

An alternative intriguing but less well researched way to implement obstacle avoidance is using end-to-end deep learning to capture the obstacle avoidance skill of a human operator. One advantage here is the ability to use cheap and power-efficient cameras – or in fact arbitrary sensors – for obstacle avoidance rather than more commonly used laser-range sensors. This is the approach followed by [12]. However, they found that such end-to-end trained deep learning models do not perform as well at their task as specialized sensors with appropriate algorithms. Also, the need for large amounts of human-generated training data makes building and adapting such systems relatively costly and complex.

1.1 Scientific Contribution

Here, we propose a new incremental training method – inspired by train-on-error (TOE, well-known from spam filtering e.g. [11]) – that uses bumper sensors to detect collisions and automatically generates end-to-end training data, which is then used to improve the initial model stepwise towards a smaller collision rate. This method is very efficient and needs only a small proportion of computational resources for each incremental step – on the order of 0.7% to 6.4% of a full training – while yielding consistent improvements on collision rate. At present, this process is semi-automated and needs only minimal human intervention. However full automation is clearly feasible with some additional work.

The method can in principle be applied to any robot with cameras and bumper sensors that has sufficient computational and memory resources to store locally or send for processing a small number of recorded camera frames, provided that an initial bootstrapping deep learning model for obstacle avoidance already exists. With sufficient but achievable computational resources on the robot, even a local on-board incremental retraining may be feasible.

Our initial goal is not to compete with well-established obstacle avoidance algorithms such as VFH* [14] or DWA [4] which would likely already perform very well here and which we could implement with modest extensions of our robot platform. Rather, we primarily aim to achieve similar obstacle avoidance performance as these well-established systems but using a trained deep-learning model with cameras as its only input. Such a system may then be used to train obstacle avoidance settings which are not yet well supported by state-of-the-art system in exactly the same way. This will also tell us whether high-performance obstacle avoidance for small robots using only cameras is generally feasible, how complex and large a deep learning network needs to be to achieve this, and how long it will need to be trained and in which way. We see this as a complementary work to currently popular deep learning reinforcement systems.

The current limitation of our proposed approach is that initially a pretrained end-to-end deep learning model is still needed, so a certain amount of human-generated training data is essential. In Discussion we note some simpler models which do not rely on this and which may be used instead. The added value of our method is that an existing pretrained model can be semi-automatically improved or adapted to a new environment or robot platform in a very cost-effective way without requiring new human-generated training data and with only moderate cost in retraining due to its incremental nature. This is in contrast to currently popular deep reinforcement learning systems which are very costly even in their incremental versions and additionally need extensive simulations which are not necessary for our approach. In our approach, training data is generated in the field so there can be no problem in transferring models from simulation to reality as in most deep reinforcement learning systems.

1.2 Outline

We will first describe related work in three different categories: end-to-end deep learning, incremental training and other approaches to obstacle avoidance. Afterwards we will shortly describe the ToyCollect open source robotics platform on which our work has been implemented and tested, and describe the relevant robot designs we have been using. Then we will describe the setup of this experiment, followed by a description of the proposed incremental training approach in detail. This is the main contribution of our paper. At the end of that section we will summarize the results. This section is then followed by a Discussion section where we will describe qualitative observation on the trained models as well as discuss ways to make the incremental training completely automated. We close with a short conclusion that summarizes our main findings.

2 Related Research

For clarity, we have divided related work into those that use end-to-end deep learning, those that use incremental retraining, and all other approaches.

2.1 End-to-end Deep Learning

[12] describe an end-to-end learning approach to obstacle avoidance and deploys it on three different robots in indoor and outdoor settings. They replicate the findings of [9, 8] to a large extent but note that performance is not yet competitive to alternative obstacle detection methods such as specialized sensors. They also introduce the ToyCollect open source hardware and software platform which we have used for this project.

[13] extends the earlier work by evaluating state-of-the-art stereo algorithms, but find them unsuitable for deployment on the robot by their performance alone. They also propose the disparity-sensitive deep learning network NCC-DISP that learns to avoid obstacle based on disparity data from stereo cameras, but note that its performance is still only comparable to the older end-to-end trained systems – and not better.

[9] describe a purely vision-based obstacle avoidance system for off-road mobile robots that is trained via end-to-end deep learning. It uses a six-layer convolutional neural network that directly processes raw YUV images from a stereo camera pair. They claim their system shows the applicability of end-to-end learning methods to off-road obstacle avoidance as it reliably predicts the bearing of traversible areas in the visual field and is robust to the extreme diversity of situations in off-road environments. Their system does not need any manual calibration, adjustments or parameter tuning, nor selection of feature detectors, nor designing robust and fast stereo algorithms.
The earlier published technical report, [8], extends the previously mentioned paper with additional experiments, a much more detailed description of the hardware setup, and a slightly more detailed description of the deep learning network. A training and test environment is described. An even more detailed description of how training data was collected is given. They found that using information from just one camera performs almost as well as using information from both cameras, which is surprising. Modified deep learning networks which try to control throttle as well as steering angle and also utilize additional sensors performed disappointingly.

[2] describe a system similar to [9] that is trained to drive a real car using 72 hours of training data from human drivers. They note that the distance between crashes of the original *DAVE* system was about 20 meters. They add artificial shifts and rotations to the training data. They report an autonomy value of 98%, corresponding to one human

intervention every 5 minutes. However, their focus is on lane following and not on obstacle avoidance. They used three cameras – left, center, right – and a complex ten layer convolutional neural network.

[10] describe an end-to-end motion planning system for autonomous indoor robots. It goes beyond our approach in also requiring a target position to move to, but uses only local information which is similar to our approach. Their approach uses a 270° laser range finder and cannot be directly applied to stereo cameras. The depth camera on our robot would in principle be usable to implement this approach, but has a horizontal FOV of only 64° and thus a factor four narrower field-of-view. Their model is trained using simulated training data and as such has some problems navigating realistic office environments.

[15] describe a convolutional neural network that learns to predict steering output from raw pixel values. They use a car driving simulator instead of real camera recordings, and they use three simulated cameras instead of our two real cameras. They explicitly address overfitting and vanishing gradient. They note several papers on end-to-end-learning for autonomous driving including [9] – however it should be noted that most of the mentioned papers are concerned with car driving and lane following, and not with obstacle avoidance, which are overlapping but different problems.

2.2 Incremental Training

[7] describe an incremental deep reinforcement learning model for navigation. It uses a baseline model which is incrementally improved by sampling from the current navigation policy which is evaluated by a loss function. They describe both simulation and real-life experiments. However, the real-life experiments are based on a simulation of the known 2D occupancy grid and thus their approach cannot be used in real-life without such an extensive simulation step and a known 2D occupancy grid. Also, their deep reinforcement learning approach is very different from our end-to-end-learning-plus-incremental training on collisions approach.

[5] describe a deep learning neural network that learns obstacle avoidance in a class room setting from human drivers, somewhat similar to our system. As starting point they use a deep learning network that has been pretrained on an image classification task and reuse some of the hidden layers for incremental training (finetuning) the network for obstacle avoidance. While technically an incremental training approach, it is still based on about two hours of human-generated training data. In ten test runs, the robot rarely grazed a chair or cardboard box, effectively demonstrating almost perfect obstacle avoidance behaviour. However, they do not propose any approach to automatically generate training data as we do. It may be feasible to combine both approaches and thus reduce or even remove the need for additional human training data when adapting to new environments. However it is unclear whether their robot includes bumper sensors which may be necessary for safe collision detection.

[6] describe a combined robot system of ad-hoc code with a pretrained deep learning network that successfully tracks and follows a second robot in two settings while also avoiding obstacles. They mainly focus on tracking and not on obstacle avoidance as we do here. Tracking uses a modified YOLO deep learning network for initial slow object detection and a kernel correlation filter for continuous fast tracking. Obstacle

Table 1. Robot Hardware Overview

Type	Robot Base	Motors	Motor controller	Camera and Sensors	Controller	Chassis	Power	LocalDL
v1.3 (*K3D*)	Pololu Zumo (#1418)	2x Pololu 100:1 brushed motor (#1101)	Pololu Qik 2s9v1 Dual Serial (#1110)	1x RPi Camera Rv1.3 w/ Kúla3D Bebe Smartphone 3D lens[1]	1x RPi 3B+	Modified transparent chassis[2]	4x 3.7V 14500 LiIon	Yes 8fps
v1.21 (*R2X*)	Pololu Zumo (#1418)	2x Pololu 100:1 brushed motors (#1101)	Pololu Qik 2s9v1 Dual Serial (#1110)	2x RPi Camera Rv2.1	2x RPi ZeroW	Three-part 3D-printed chassis[3]	4x 3.7V 14500 LiIon 4x 1.5V AA Alkaline 4x 1.2V AA NiMH	No $<$1fps
v2.2 (*OUT*)	Dagu Robotics Wild Thumper 4WD (#RS010)	4x 75:1 brushed motors (included)[4]	Pololu Qik 2s12v10 Dual Serial (#1112)	2x RPi Camera Rv2.1	1x RPi CM3 on official eval board	None (open case)	1x 7.2V 5000mAh LiPo	Yes 8fps
v1.4 (*SP1*)	3D-printed	2x Pololu 100:1 brushed motors (#1101)	Pololu Qik 2s9v1 Dual Serial (#1110)	2x RPi Camera Rv2.1 1x Intel Realsense D400 Left/right bumper sensor Magnetic motor encoders	1x RPi CM3 on StereoPi V1	Modular 3D-printed chassis	1x 3.7V LiIon 3300mAh 18650 w/o BPC[5] 8A constant discharge	Yes 8fps

avoidance relies exclusively on an active sensing depth camera, and we could in principle implement their approach using the depth camera on our robot. However, at this point we do not yet have two robots in active use.

2.3 Other Methods

[1] describe a robot system to detect obstacles a posteriori by acceleration sensor data. They propose a logistic regression model trained on known collisions as detected by a human observer, and show that it can detect new collisions in 13 out of 14 cases. However, as we try to *avoid* obstacles their approach cannot be directly applied. It would still be useful for autonomous collection of training data and online training without relying on more commonly used bumper sensors, and as complementary virtual sensor to detect rare collisions that are not detected by bumper sensors.

[14] propose VFH*, a local obstacle avoidance algorithm based on earlier work on VFH+ and VFH[6]. It works by utilizing range sensors such as ultrasound sensors, laser-range sensors or depth cameras to continually create and update a local coarse map of likely obstacle positions by distance (a Vector Field Histogram), shifts this map during robot movements, and at high frequency computes a polar histogram at the current robot position to determine possible movement directions – i.e. those where no obstacles were found within sensor range. Its performance in our relatively simple setting is likely superior to our presented method, although the relatively high minimum range of our depth camera of 30cm – almost three times the length of our robot and thus an order of magnitude higher than in their paper – and the small field-of-view of 64° might prove challenging. Dead reckoning direction might also drift as our crawler track drive tends to slip and thus the motor encoders on their own might not be accurate enough over time. However, adding a combined acceleration and gyroscope sensor should be sufficient to resolve this.

[1] A double periscope that divides the camera optical path into two halves and moves them apart using mirrors, effectively creating a stereo camera pair from one camera.

[2] Sadly, no longer available

[3] Can be plotted as one part for perfect alignment.

[4] Left two and right two are each connected.

[5] Battery Protection Circuit

[6] Vector Field Histogram

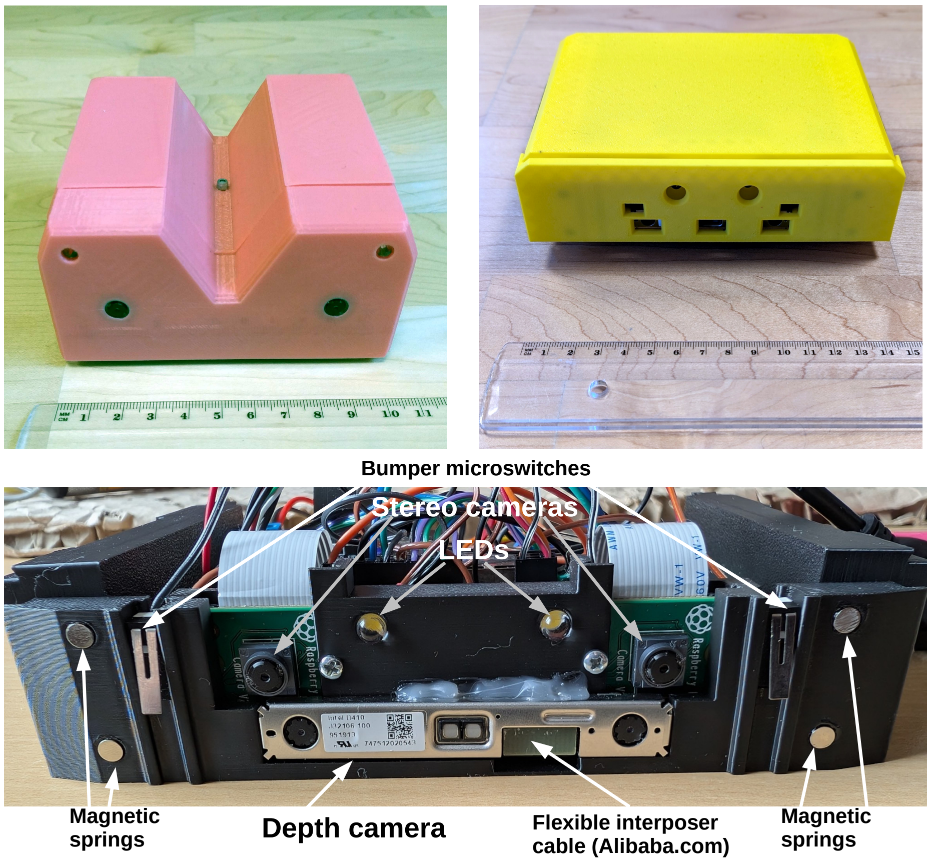

Fig. 1. Robots v1.21 (*R2X*), v1.4 (*SP1*) top left to top right, ruler units: cm. Size as width/depth/height: *R2X* = 10.5x10x6 cm^3, *SP1* = 15.5x12.5x5 cm^3. Below: Front of *SP1* after removing bumper.

[4] propose DWA[7], which also utilizes range sensors and creates a local map of obstacles similar to the previous approach. However the obstacles are represented as *obstacle lines* rather than as a vector field histogram which is perhaps less suitable for our complex environment. They propose motion equations and approximations for a synchro-drive robot, and use this to restrict the permissible velocities yielding an efficient algorithm. The same caveats as in the previous approach also apply here: the minimum range and small field-of-view of the depth camera might be problematic, as would be dead-reckoning using only motor-encoder input. We will still consider both approaches for future work, but again note that this algorithm is probably sufficient for almost-no-collision obstacle avoidance in our test environment and therefore a direct comparison is currently not useful.

[7] Dynamic Window Approach

3 ToyCollect Platform

All experiments were conducted with our ToyCollect robotics open source hardware/-software platform (`https://tc.seewald.at`). A hardware overview of all our robots can be found in Table 1 while Fig.1 shows images for robot v1.4 (*SP1*) and its precursor, v1.21 (*R2X*). For the following experiments we used the *SP1* robot which combines stereo cameras, depth camera[8], magnetic motor encoders and bumper sensors and was slightly lower than the older robot v1.21 *R2X* albeit about 50% wider.[9] The main motivation for building this new robot was to have a platform as small as *R2X* but capable of simple local deep-learning processing and with additional sensors to enable (semi-)autonomous testing and obstacle avoidance.

The *SP1* robot is the first one with a completely modular 3D printed design. It consists of a baseplate that contains the StereoPi V1, the RPi Compute Module 3, voltage converter, motor controller and two fans (not shown), as well as stereo cameras, a depth camera[8], two bumper sensors, four magnetic springs and two leds (see Fig. 1 lower image); two crawler tracks left/right which contain the motors, wheels and tracks; a mechanical front bumper with magnetic springs; and a slideable top to close it all up.

We added the following three avoidance behaviours to enable more autonomous tests. The first two are automatic reflexes, the third one must be activated externally.

1. **Bumper sensor:** An activation of the bumper sensor (either left or right) reverses the robot for about one second and then turns it into a random direction at least 90° away from the obstacle.
2. **Magnetic motor encoders:** Should both motors be blocked more than two seconds as measured by the magnetic motor encoders, the same avoidance behaviour as above is started.
3. **Bluetooth remote:** Should the robot become stuck in a way that is not detected by 1. or 2., an override can be pressed on the remote which starts the same avoidance behaviour as before.

In behaviour 3., we could always override the robot using a bluetooth remote. However, this was only necessary on average every 7 minutes when the robot was wedged against an obstacle with still moving wheels without active bumper sensors.[10] In that case, we manually forced avoidance behaviour by using the bluetooth remote to stop the robot for at least 5 seconds. This then started the avoidance behaviour 2. due to blocked motors. We separately tracked this case but sensibly counted it also as a collision, albeit in that case – and for behaviour 2. – no training data can be generated as the avoidance driving direction is not known (see below).

4 Experimental Setup

As the very first step, we provided a fixed test environment with cabinets, book shelves, tables and chairs – and one ergometer – at specific positions, and also defined the robot

[8] Not used within this work.

[9] We put the StereoPi V1 *between* the crawler tracks as otherwise the height would have been much greater than for the older robot design *R2X*.

[10] This happened most often when running against the tilted feet of the rectangular chairs.

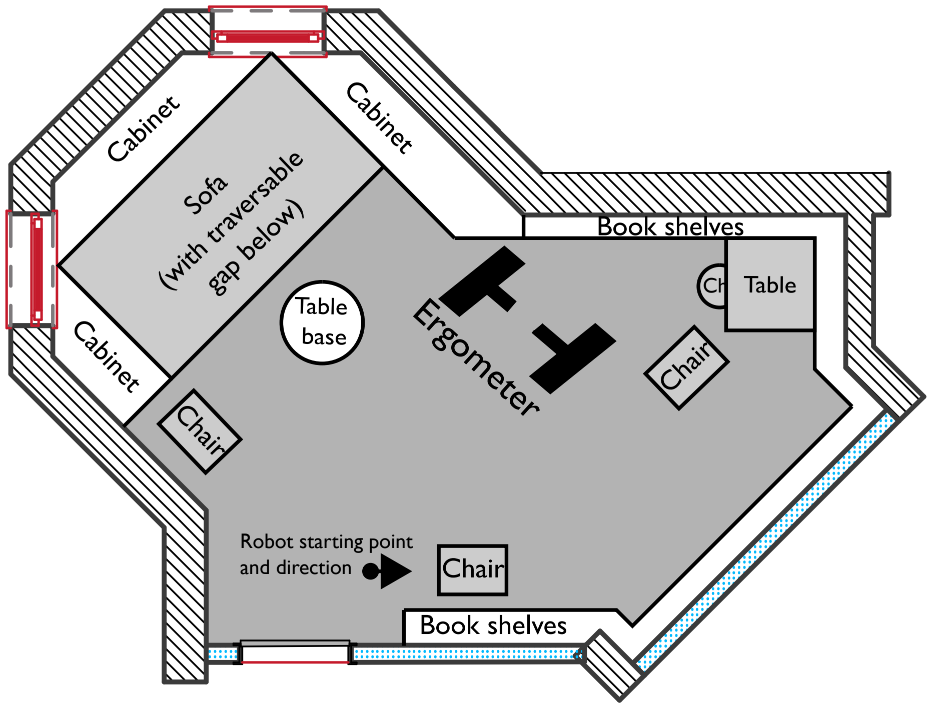

Fig. 2. Fixed test environment with about 14.5 m^2 traversable area. The small round chair is a children's office chair with soft wheels. The three rectangular chairs have four wooden feet each which are tilted at about 30° against vertical. The floor is maple parquet and quite slippery and also shiny, it often shows reflections of the bookshelfs in front of the robot. Only light and dark gray areas are traversable by the robot. It can also go below the sofa, below the table, and below each chair (except for table feet and chair feet which are of course obstacles). It can go below the ergometer at one place in the middle since there is just enough space. It cannot traverse the table base which is too high.

starting point and direction. Fig. 2 shows the environment in detail. It had about 14.5 m^2 traversable area from a room total of 17.8 m^2.

We first deployed the original obstacle avoidance model (*indoor/Cl3*) described in [12] and originally trained on *R2X* using about 10h of human-generated training data to the new robot *SP1*. As input the stereo-camera images were each scaled to 149x58 resolution via linear interpolation and split into equal-sized Y, U and V channels, yielding six channels, i.e. a (None,149,58,6) tensor. As output three classification output units for left, straight forward, right were defined, yielding a (None,3) tensor. The network had 88,543 trainable parameters. For the exact structure of the network we refer to the original paper.[11]

However, performance was suboptimal since the *SP1* robot is about 50% wider as the *R2X* robot – where the data for this model was collected – so we observed many

[11] [12], page 6, start of Results.

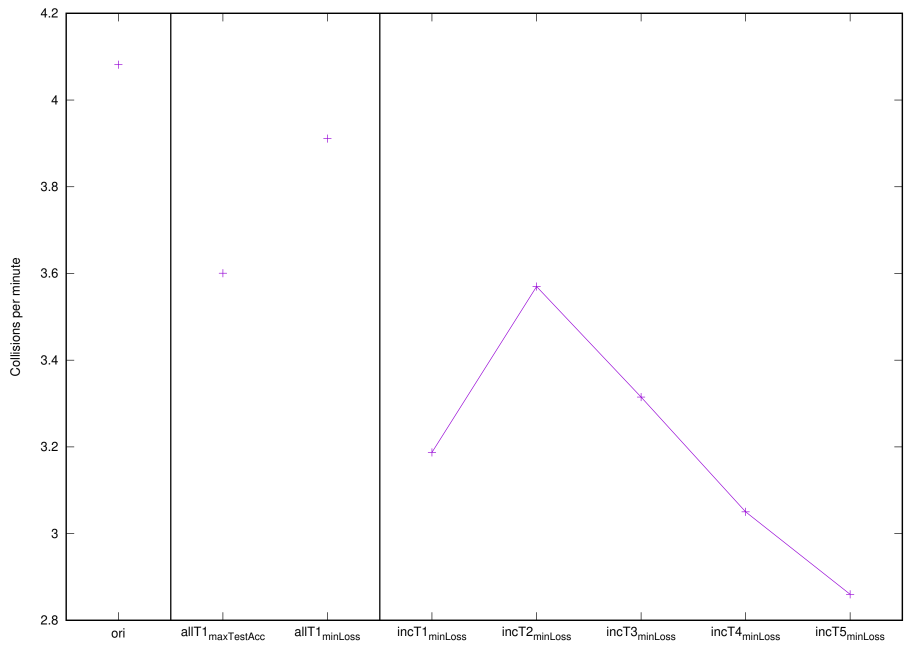

Fig. 3. Results from obstacle avoidance experiments on robot *SP1*. Each data point is based on an about 95min continuous run within the same environment. All are stereo camera deep learning models of the same type and size as in [12], but differently trained. *ori* = original model described in [12]. $allT1$ = original training data from [12] plus $incT1$ training data, best model by *minLoss* or *maxTestAcc*. $incT1-5$ = incrementally trained – for more details see text.

avoidable collisions. In a way, it could be discerned that the model implicitly assumed a narrower robot and was driving accordingly.

Unfortunately it was not possible to determine the performance of the original obstacle model on the *R2X* robot within the fixed test environment because i) the old robot platform did not have bumper sensors and thus could not detect and measure collisions automatically, and ii) *R2X* did not have sufficient computational resources to run the deep learning model locally, so it would have been necessary to stream the stereo images via Wifi to a central processing server – which gives comparable frame rates but a much higher latency and would therefore also have distorted the results.

Next, we intended to incrementally train the original model for better performance. For this, we took the original model and continued its training incrementally with newly generated data. To generate new training data, we considered each bumper collision from a 95min[12] test run of the robot within the test environment (see Fig. 2), took the 2s of frames before each collision, and used as driving direction the one opposite the

[12] Effectively one battery charge from starting the robot until it stops due to low battery, so it might be a bit shorter or longer... more precisely, 5,748.33 ± 240.23 seconds.

Table 2. Results of obstacle avoidance experiments

Run	#Sam. trained	Coll. #Sam.	Train #Steps	Coll. per min.
ori	70,745	3,519	267.9k	4.08
$allT1_{maxTestAcc}$	74,264	n/a	337.8k	3.6
$allT1_{minLoss}$	74,264	n/a	7.2k	3.91
$incT1_{minLoss}$	3,519	2,828	17.2k	3.19
$incT2_{minLoss}$	2,828	2,933	3.4k	3.57
$incT3_{minLoss}$	2,933	2,679	4.9k	3.31
$incT4_{minLoss}$	2,679	2,343	2.8k	3.05
$incT5_{minLoss}$	2,343	n/a	1.8k	2.86

bumper sensor[13], yielding 3,519 additional training stereo images with directions. This new training data was used to incrementally train the original model for another 400k steps, and the model with the maximum test set accuracy (62.01% at steps=330.6k) as well as the model with the minimum loss[14] (2.33 at steps=7.2k) were both tested for the length of one battery charge (about 95min) in the same environment. Fig. 3 and Table 2 shows both these results as $allT1_{minLoss}$ resp. $allT1_{maxTestAcc}$. Collisions are reduced in both cases.

Initially we had intended to use depth camera measurements to determine the best avoidance direction. However, this did not work.[15] Due to this we simply determined avoidance direction after collisions by referring to the bumper sensors as mentioned above, i.e. moving away from the activated bumper sensor.

5 Incremental Training

We investigated three variants for stepwise incremental training – in each case only using the new data obtained from collisions in the previous run for training, but with different test data. The main issue here was to decide when to stop training to prevent both overfitting the relatively small collision data, as well as to prevent changing model

[13] I.e. maximum left (-127) for right bumper activation, and maximum right (+127) for left bumper activation.

[14] We should perhaps note that this is obviously also test set loss and not training set loss.

[15] To test for feasibility, we implemented a simple obstacle avoidance algorithm solely based on measured depth data, ignoring the stereo cameras completely and also without reference to the trained deep learning model. We implemented a simple search algorithm to determine rectangles within the depth image where the measured distance was at least 0.3 meters – ignoring points where no measurement existed – and which were of sufficient size so that the robot could drive through. We then computed drive direction to the center of the largest rectangle. However, results were even slightly worse than the original stereo deep learning model, probably because the depth camera could not measure any distance below about 30cm.

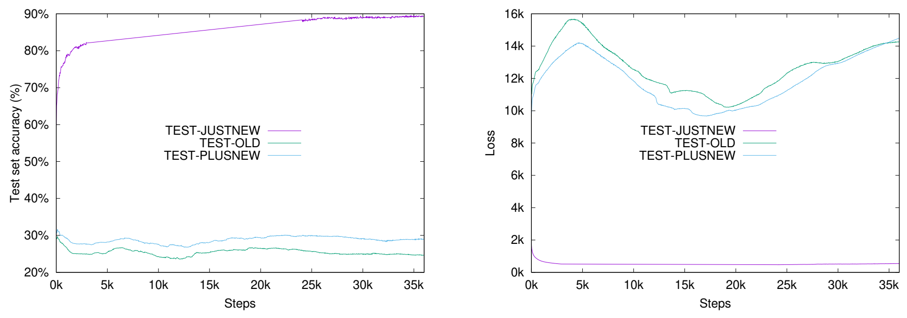

Fig. 4. Learning curves for the first collision training set (obtained by running the *ori* model in the test environment). Test data is either collision test data (TEST-JUSTNEW), the *ori* model original test data (TEST-OLD), or both (TEST-PLUSNEW). Steps start at the original model with 0.

weights too much as this may lead to too simple behaviour only accounting for collision avoidance.[16]

1. **Using just the collision test data (TEST_JUSTNEW):** This gives learning curves with very weak local minima, in fact both loss and test set accuracy improve for a very long time. Also, it is likely that the given model is prone to overfitting with such a small dataset of just a few thousand samples.[17]
2. **Using just the original test data (TEST_OLD):** This gives sensible learning curves for loss, but very noisy learning curves for test set accuracy. Additionally, this does not include the new test data at all – here we just use it for training – so it may be hard to interpret the obtained losses and accuracies, and determine when the new samples have been sufficiently well learned. We felt that this may give undue importance to the old training data at cost of the new collision data that we would like to be learned.
3. **Using the original test data plus the new collision test data (TEST_PLUSNEW)** This yields very similar curves as in 2., however now includes both the original and the new test data – albeit with a much higher weight on the original test data – and we have therefore used it for all our experiments.

Fig. 4 shows the learning curves for all three variants and for both loss and test set accuracy. Based on this we used the third variant (*TEST_PLUSNEW*) and used minimum loss to choose the best model rather than the more commonly used maximum test set accuracy. The latter was due to the following considerations.

Firstly, the test set accuracy is not a good indicator for true collision avoidance performance as in many cases an obstacle can be avoided on the left as well as on the right and the model may switch between directions from frame to frame, sometimes forcing a collision (as mentioned in [12], Ch. 6, second paragraph). This is corroborated later when we observe a qualitatively and quantitatively better avoidance behaviour at about half the test set accuracy of the original model.

[16] Such as always driving in circles, thus effectively preventing all collisions.
[17] See Table 2 for more precise dataset sizes.

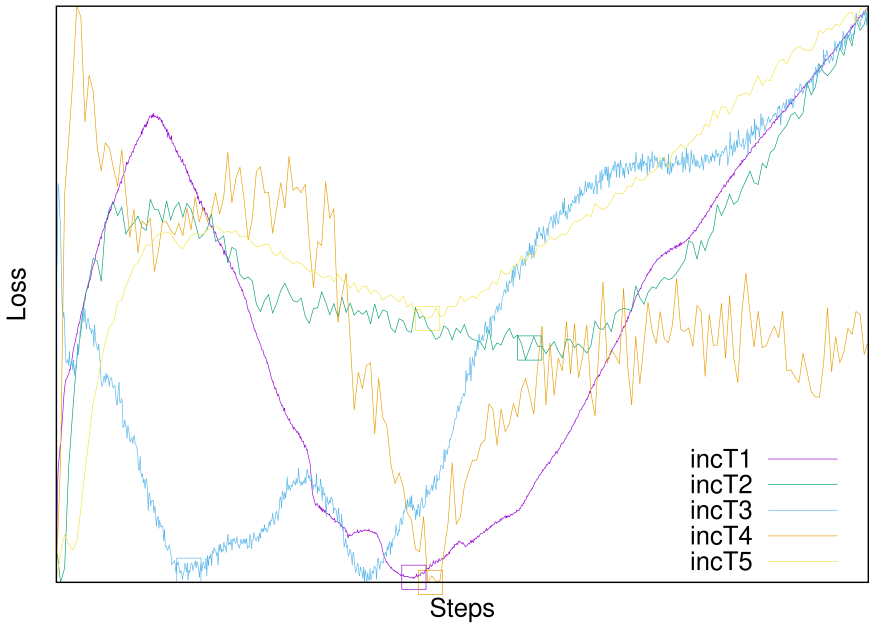

Fig. 5. Learning curves for all five *incT* runs. Since we only focus on curve shape here, all curves were separately and arbitrarily scaled on both axes to make them fit exactly into this graph. Thus absolute loss and steps are not observable here and we therefore removed the units on both axes. The local minima chosen for the next cycle are marked by squares in the same color as the learning curves, centered on the chosen point. Note that the $incT5$ minimum is marked but has not been used yet.

Secondly, the test set accuracy does not lend itself to a clear stopping criterion as it changes quite often. Loss, however, had a clear pattern with increasing loss, then decreasing, and later increasing again, which lent itself to cutting off at the first – or in some cases the second – local loss minimum. Also, the minimum loss models were always found within 5,000 training steps (except for $incT1_{minLoss}$), thus incurring only a very small training effort.[18]

The generation of the collision data as described above enforces a consistent direction as all frames observed at most 2s before collision receive the same drive direction.

As test data *TEST_PLUSNEW* we always used the original test data (*ori*) plus only the newest incremental test data. This was motivated from an initial experiment where the learning curves did not show a clear loss pattern when combining two incremental test data sets with the original test data (data not shown). However since the old test data has already been used to choose the model, this choice can be seen as a logical extension of the initial decision to not reuse any old data.

[18] In fact even on our very old NVidia GTX 1080 TI Founders edition GPU, we could train one collision dataset in about two hours.

The incremental training was therefore conducted as follows. For $incT1_{minLoss}$ we took the collision training data that was collected during the previous run of the original model (see run *ori* in Table 2, column *Coll(ision) #Sam(ples)*), split it into 80% training and 20% test, and trained it on the original model incrementally, using only the collision training data but both the *ori* test set and the collision test set (i.e. *TEST_PLUSNEW*), choosing the model with minimum loss after the first initial loss increase but before the second initial loss increase (i.e. the first – in one case the second – local loss minimum). It will turn out that this choice was always well defined also for the consecutive steps as seen in Fig. 5 which shows the loss learning curves for all $incT1-5$ experiments.

This new model was then converted to TensorflowLite format, deployed to the robot, and run in the same physical environment (see Fig. 2) for a similar time, yielding a second collision training dataset which we used to incrementally train the $incT2_{minLoss}$ model similarily as above (starting at the previous $incT1_{minLoss}$ model), and so on. The advantage is that we train on error but keep much of model largely unchanged so that we expect the collisions to decrease without incurring overfitting or degenerated movement patterns (such as always turning in circles) – and this is exactly what we observed.

5.1 Results

Fig. 3 and Table 2 show the final results of these experiments as well as the total number of samples used for training and testing each model. As can be seen, the general tendency from $incT1$ to $incT5$ is for the collisions to go down. The only outlier is $incT1$ itself. We speculate that it might be due to the many collisions caused by the original model assuming the robot to be narrower, which was corrected after the more complex and longer training there, and the consecutive incremental training steps then corrected other types of collisions which were more diverse and therefore harder to learn, yielding an initial increase in collision rate from $incT1$ to $incT2$. Still, from $incT2$ onward the collision rate is always decreasing.

6 Discussion

One additional advantage of the semi-autonomous retraining that we found is its usefulness to adapt a pretrained model to changed robot characteristics. For example, the original model was trained on v1.21 (*R2X*), a much narrower robot, using about 10 hours of training data obtained by manually driving the robot through a variety of environments without collisions.[19] Normally it would have been necessary to again collect a similar amount of training data on the new robot. With our method, this was not necessary. In fact there may be some slight indication that the first large reduction in collision rate (about 22% from *ori* to $incT1_{minLoss}$) is driven by the adaption of the model to the new robot platform and further improvements focus on other aspects of driving, yielding a slight increase in collision rate after the first large reduction. It should be noted that both the vertical distance to the ground and distance between the stereo cameras remained unchanged between both robot designs.

[19] For more details see [12].

Apart from the quantitative evaluation, we also observed emergent qualitative behaviour patterns as follows.

– **Wall-following:** The robot would follow a wall for a certain distance and then veer away from it.
– **Rubbing against obstacles:** The robot would move towards an obstacle and veer away at the last minute, often rubbing against the obstacle's side without activating the bumper sensor.
– **Fixed routes:** We observed the robot taking the exact same complex route up to three times in a row. However, the total time that the robot was staying in these loops was still negligible.
– **Scanning behaviour:** Where with the original model *ori* the robot would follow clear paths with seemingly clear intention – sometimes directly towards an obstacle – the modified models $incT1-5$ show a more cautious approach: often the robot will move slightly to one side, then slightly to the other as if it is looking around or it is not sure where it wants to go.

The changed behaviour patterns in fact introduce an almost biological aspect into the movements of the robot. It should be noted that we are using a very small deep learning model with less than 90,000 trainable parameters, so perhaps these effects are more pronounced because of that.

Further work is needed to enable completely autonomous testing and automatic generation and evaluation of obstacle avoidance data and models, which is currently only a semi-automated process. The most important steps to be taken to achieve this are as follows.

1. Currently, the robot top and the battery must be removed and the battery must be charged separately. The robot should optimally be self-charging and have a charging station, either inductively or via a conductive plate.
2. On average about every 7 minutes, the robot wedges itself below a chair foot and gets stuck without either bumper sensors or magnetic motor encoders detecting an obstacle. A redesign of the bumper plate, less slippery floors, or increasing the weight of the robot – or adding an acceleration sensor and implementing [1]'s obstacle detection algorithm – could be used to address this.[20]
3. We can not at present automatically determine whether the obstacle avoidance model only visits part of the room or whether it even just runs along fixed loops. For this, an exact position of the robot would be useful. This could be obtained by implementing a SLAM [3] algorithm on the robot. However due to the complexities of setting up SLAM, an alternative tracking method using a ceiling camera and color markings on the robot top would be preferable.[21]

At present our approach still relies on a human-trained end-to-end model for bootstrapping. This is a slight limitation as the final achievable performance may thus rely on

[20] We could also switch the chairs against ones without tilted feet or use a room with less slippery floors but that would be cheating.

[21] Although the latter could of course not track robot position when it goes below tables, chairs or the sofa, this could be resolved by dead-reckoning.

the quality of human training data used to train the initial model. However it may be feasible to replace this with a trivial model (e.g. driving just straight ahead) or a random walk model while still keeping the same improvements. This could be done by collecting data generated by the robot while running one of these alternative models – in this case excluding collisions – and thus to create an initial training set. More work is needed to determine whether this approach is feasible.

7 Conclusion

We proposed an incremental training method inspired by train-on-error for end-to-end obstacle avoidance in mobile robots, and demonstrated that it could reduce the collision rate consistently with just a few incremental training steps and without requiring additional human-generated training data. It both performs better and is much faster to train than the more common approach to completely retrain the model from scratch, and also improves obstacle avoidance behaviour qualitatively. We see our approach as complementary to currently popular deep reinforcement learning systems which are very costly even in their incremental versions and additionally need extensive simulations with the corresponding problem in transferring models from simulation to reality. This is not an issue with our approach as all data is generated in reality on the robot, so no simulation is needed.

While not fully automated at this time, it is feasible to add this with modest effort (see Sec. Discussion). A fully automated system would also allow to more easily test the approach on a diverse set of test environments rather than the one fixed test environment we used here.

Acknowledgments. We would like to thank Lukas D.-B. for 3D chassis designs of robot *R2X* and *SP1*. We would also like to thank all students that have worked on the project all these years.

Disclosure of Interests. The author declares no competing interest.

References

1. Becker, F., Ebner, M.: Collision detection for a mobile robot using logistic regression. In: Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,. pp. 167–173. INSTICC, SciTePress (2019). https://doi.org/10.5220/0007768601670173
2. Bojarski, M., Del Testa, D., Dworakowski, D., et al.: End to end learning for self-driving cars. Tech. Rep. 1604.07316, Cornell University (2016)
3. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.: Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016)
4. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE robotics & automation magazine **4**(1), 23–33 (2002)
5. Khan, M., Parker, G.: Vision based indoor obstacle avoidance using a deep convolutional neural network. In: Proceedings of the 11th International Joint Conference on Computational Intelligence - NCTA, (IJCCI 2019). pp. 403–411. INSTICC, SciTePress (2019). https://doi.org/10.5220/0008165104030411

6. Luo, W., Xiao, Z., Ebel, H., Eberhard, P.: Stereo vision-based autonomous target detection and tracking on an omnidirectional mobile robot. In: Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,. pp. 268–275. INSTICC, SciTePress (2019). https://doi.org/10.5220/0007835702680275
7. Luong, M., Pham, C.: Incremental learning for autonomous navigation of mobile robots based on deep reinforcement learning. Journal of Intelligent & Robotic Systems **101**(1), 1 (2021)
8. Muller, U., Ben, J., Cosatto, E., Fleep, B., LeCun, Y.: Autonomous off-road vehicle control using end-to-end learning. Tech. rep., DARPA-IPTO, Arlington, Virginia, USA (2004), aRPA Order Q458, Program 3D10, DARPA/CMO Contract #MDA972-03-C-0111, V1.2, 2004/07/30.
9. Muller, U., Ben, J., Cosatto, E., Fleep, B., LeCun, Y.: Off-road obstacle avoidance through end-to-end learning. In: Advances in neural information processing systems. pp. 739–746 (2006)
10. Pfeiffer, M., Schaeuble, M., Nieto, J.I., Siegwart, R., Cadena, C.: From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. CoRR **abs/1609.07910** (2016), `http://arxiv.org/abs/1609.07910`
11. Seewald, A.K.: An evaluation of naive bayes variants in content-based learning for spam filtering. Intelligent Data Analysis **11**(5), 497–524 (2007). https://doi.org/10.3233/IDA-2007-11505, `https://journals.sagepub.com/doi/abs/10.3233/IDA-2007-11505`
12. Seewald, A.K.: Revisiting End-to-end Deep Learning for Obstacle Avoidance: Replication and Open Issues. In: Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020), Valetta, Malta. pp. 652–659 (2020)
13. Seewald, A.K.: Evaluating two ways for mobile robot obstacle avoidance with stereo cameras: Stereo view algorithms and end-to-end trained disparity-sensitive networks. In: Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022), Online Streaming. pp. 663–672 (2022)
14. Ulrich, I., Borenstein, J.: VFH*: Local obstacle avoidance with look-ahead verification. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). vol. 3, pp. 2505–2511. IEEE (2000)
15. Wang, Y., Dongfang, L., Jeon, H., Chu, Z., Matson, ET.: End-to-end learning approach for autonomous driving: A convolutional neural network model. In: Rocha, A., Steels, L., van den Herik, J. (eds.) Proc. of ICAART 2019. vol. 2, pp. 833–839 (2019)