Evaluating Two Ways for Mobile Robot Obstacle Avoidance with Stereo Cameras: Stereo View Algorithms and End-to-End Trained Disparity-sensitive Networks

Alexander K. Seewald
Seewald Solutions, Lärchenstraße 1, A-4616 Weißkirchen a.d. Traun, Austria

Keywords: Deep Learning, Obstacle Avoidance, Autonomous Robotics, Mobile Robotics, View Disparity, Stereo View Algorithms.

Abstract: Obstacle avoidance is an essential feature for autonomous robots. Recently, stereo view algorithm challenges have started to focus on fast algorithms with low computational expense, which may enable obstacle avoidance on mobile robots using only stereo cameras. Therefore we have evaluated classical and state-of-the-art stereo salgorithms qualitatively using internal datasets from Seewald (2020), showing that – although improvements are discernable – current algorithms still fail at this task. As it is known (e.g. from Muller et al. (2004)) that deep learning networks trained on stereo views do not rely on view disparity – confirmed by the fact that networks perform almost equally well when trained with only one camera image – we present an alternative network which is end-to-end trained on a simple layer of biologically plausible disparity-sensitive cells and show that it performs equally well as systems trained on raw image data, but must by design rely on view disparity alone.

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, Cadena et al. (2016)). 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.

Recently stereo algorithm challenges such as the Robust Vision Challenge[1] have focussed on robust vision of the kind that may be useful to mobile robots as an alternative for near obstacle avoidance that does not rely on combining specialized sensors or map-building.

Here, we have qualitatively evaluated classical as well as state-of-the-art stereo algorithms on internal datasets from our ToyCollect robot platform (Seewald (2020)). These datasets only include stereo views and thus cannot be quantitatively evaluated but are representative of the type of stereo views commonly obtained with mobile robots. Our qualitative evaluation indicates that despite the excellent results of state-of-the-art stereo algorithms on high-quality benchmark stereo views, these results transfer only to a very small extent to real-life stereo views.

We speculate that this is due to the algorithms determining distance by occlusion, linear perspective, size perspective, distribution of shadows and illumination, and familiar size (Kandel et al. (2000), p. 558f) – all of which are likely to differ between benchmark datasets and our internal datasets – and rely only to a limited extent – if at all – on view disparity. This is surprising as the human visual system relies very strongly on view disparity for near obstacle detection. Further support for this comes from end-to-end deep-learning models for obstacle avoidance trained directly on stereo views (Muller et al. (2004)), where it is known that models trained using information from just one camera perform almost as well as models trained using information from both cameras.

[1] https://robustvision.net

Seewald, A.
Evaluating Two Ways for Mobile Robot Obstacle Avoidance with Stereo Cameras: Stereo View Algorithms and End-to-End Trained Disparity-sensitive Networks.
DOI: 10.5220/0010878500003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 663-672
ISBN: 978-989-758-547-0; ISSN: 2184-433X

Table 1: Robot Hardware Overview.

Type	Robot Base	Motors	Motor controller	Camera	Controller	Chassis	Power	LocalDL
v1.3 (*K3D*)	Pololu Zumo (#1418)	2x Pololu 100:1 brushed motors (#1101)	Pololu Qik 2s9v1 Dual Serial (#1110)	1x RPi Camera Rv1.3 w/ Kúla3D Bebe Smartphone 3D lens[2]	1x RPi 3B+	Modified transparent chassis[3]	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[4]	4x 3.7V 14500 LiIon *or* 4x 1.5V AA Alkaline *or* 4x 1.2V AA NiMH	No <1fps
v2.2 (*OUT*)	Dagu Robotics Wild Thumper 4WD (#RS010)	4x 75:1 brushed motors (included)[5]	Pololu Qik 2s12v10 Dual Serial (#1112)	2x RPi Camera Rv2.1	1x RPi CM3 on official eval board	None (open case)	1x 7.2V 2S 5000mAh LiPo	Yes 8fps

As the lack of depth maps in our internal datasets prevents us from retraining state-of-the-art stereo algorithms, we have instead created an end-to-end trained deep learning network which is forced to learn from view disparity by mapping the network input from raw image data to a simple approximation of disparity-sensitive cells known to exist in monkey visual cortex (Kandel et al. (2000), p. 562 and Fig. 28-14). We then proceed to show that such a network performs as well at the obstacle avoidance task as the original end-to-end-trained model using raw stereo view images presented in Seewald (2020). Limitations in the used hardware and software platforms prevent us from testing the new model directly on our robots for now, so we leave that part for future work.

2 RELATED RESEARCH

As far as we know, no previous qualitative evaluation of stereo view algorithms in a mobile robotics setting for obstacle avoidance has been done. Neither are we aware of any attempt to explicitly force end-to-end trained deep learning networks to rely on view disparity for obstacle avoidance. So we will instead describe earlier attempts at obstacle avoidance here.

Muller et al. (2006) 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. They note some important points w.r.t. training data collection which we have also noted and applied for our data collection:

- High diversity of training grounds.
- Various lighting conditions.
- Collect sequences where robot starts moving straight and then moves left/right around obstacle.
- Avoid turns when no obstacles are present.
- Include straight runs with no obstacles nor turns.
- Turn at approx. same distance from an obstacle.

The earlier published technical report, Muller et al. (2004), 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.

Bojarski et al. (2016) describe a system similar to Muller et al. (2006) 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 values 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.

Pfeiffer et al. (2016) 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. However, their approach uses a 270° laser range finder and cannot be directly applied to stereo cameras. Their model is trained using simulated training data and as such

[2] 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.

[3] Sadly, no longer available

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

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

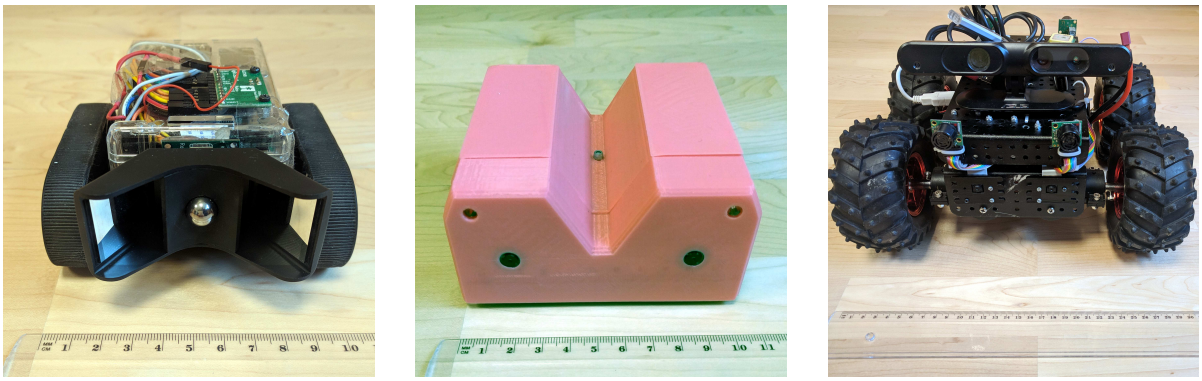


Figure 1: Robots *K3D*, *R2X*, *OUT* (left to right, ruler units: cm).

Figure 2: FOV comparison between robot *R2X* and *K3D*.

has some problems navigating realistic office environments.

Wang et al. (2019) describe a convolutional neural network that learns to predict steering output from raw pixel values. Contrary to our approach, 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 which may reduce the achievable performance also in our case.[6] They note several papers on end-to-end-learning for autonomous driving including Muller et al. (2006) – 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.

Khan and Parker (2019) 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 trained on an image classification task and reuse some of the hidden layers for incremental training. However, their approach uses only one camera and cannot be directly applied to stereo cameras. Still, their results seem promising and will be considered for future experiments.

Luo et al. (2019) 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. 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 avoidance relies exclusively on an active sensing depth camera – which we also plan to integrate into our platform in the near future – and is not separately evaluated.

Becker and Ebner (2019) 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 retraining without relying on more commonly used bumper sensors.

[6]Especially for the smaller *OUT_Train* dataset.

3 ToyCollect PLATFORM

All data was collected with our ToyCollect robotics open source hardware/software platform (https://tc.seewald.at). A hardware overview of the three utilized robots can be found in Table 1 while Fig.1 shows images for each robot with rulers for scale.

While robots *OUT* and *K3D* need only a single main controller and have sufficient computational power to run a small deep-learning model at interactive frame rates (around 8 frames-per-second) – thus enabling onboard processing – the need for two cameras on *R2X* necessitates the usage of two controllers, each connected to one camera, and each streaming the frames independently to a processing server.

OUT and *R2X* each use two cameras with a field-of-view of 62.2° horizontal and 48.8° vertical, however for *K3D* we used the older camera which has only 54° horizontal and 41° vertical field-of-view and the horizontal viewing angle is approximately halved again by the 3D smartphone lens. Fig.2 shows the difference in field-of-view between *K3D* and *R2X*.

OUT also includes a depth camera, GPS module, a 10-DOF inertial measurement unit including an accelerometer, gyroscope, magnetometer and a barometer for attitude measurement as well as a thermometer, and four ultrasound sensors. However these were not used for our experiments.

R2X includes two high-power white LEDs to allow operation in total darkness and whose brightness can be controlled in 127 steps. However these also were not used for our experiments.

Unfortunately it is not possible to connect two cameras to a RPi controller since the necessary connections are only available on the chip and not on the PCB. Only the ComputeModule allows to directly connect two cameras; however the official evaluation boards for the RPi compute module are too large to fit on the small robot. So for robot *R2X* we integrated two RPi Zero Ws into a single robot chassis and connected each to a separate camera.[7] However since the RPi Zero W is based on the original RPi 1, it is much too slow for online deep learning model processing and achieves less than 1 frames per second. So for this robot we must stream the video frames to a second more powerful plattform.

[7]In the meantime other options have become available, e.g. StereoPi, which we are currently evaluating.

4 DATA COLLECTION

We collected two different types of data (consisting of frames and speed/direction control input[8]) for indoor and outdoor obstacle avoidance. In each case we aimed for a consistent avoidance behaviour roughly at the same distance from each obstacle similar to that described in Muller et al. (2006) (see also Sec.2). However, instead of collecting many short sequences, we collected large continuous sequences and afterwards filtered the frames with a semi-automated approach. All data was collected by students during summer internships in 2017, 2018 and 2019. The students were made aware of the conditions for data collection, and were supervised for about one fifth of recording time to ensure compliance.

The collected datasets can be found in Table 2. *R2X_Train* and *OUT_Train* were those used for training the correspondingly named robots in Seewald (2020).

For *indoor* obstacle avoidance (robots *R2X*, *K3D*) we initially collected 557,640 frames in a variety of indoor and outdoor settings with a small set of small mobile robots. For training we restricted them to 267,617 frames recorded by the latest robot in 2019 to ensure all recordings were done with the same cameras which were later also used for evaluation. Stereo views were collected directly onboard *R2X* robots in uncompressed YUV 4:2:2 format on SD cards in 640x368 resolution at 10fps. Control of the robot was via paired Bluetooth controller. We first inspected the recorded sequences manually and removed those with technical issues (e.g. no movement, cameras not synchronized, test runs). Because of synchronization issues, the first minute of each sequence (up to the point when each MASTER and SLAVE synchronize with an external time server[9]) had to be removed. Additionally, sequences with very slow speed and with backward movement (negative speed) were removed along with 50ms of context. Since both cameras were independently recorded, we also removed all frames without a partner frame that is at most 50ms[10] apart. We also removed frames where movement information is not available within $\pm$25ms of the average timestamp for the image frames. Lastly, we had to remove 80% of the frames with straight forward movement as otherwise these would have dominated the training set. After all these filters, 70,745

[8]Only direction (steering) is used for training.

[9]The RPi platform does not offer a realtime clock and thus suffers from significant clock drift. It would have been quite simple to synchronize local clock during MASTER/SLAVE synchronization but we failed to consider it.

[10]Half of 100ms (i.e. 10fps recording frequency)

Table 2: Datasets.

Name	Coll.by robot	Size	DL Train?	Qual. Eval?	Comments
Mixed	R2X	557,640	No	Yes	Indoors and outdoors near houses
Outside	OUT	51,735	No	Yes	Outdoors, fields, forests, small streets, walkways and trails
R2X_Train	R2X	70,745	Yes	No	Curated subset of *Mixed* and recorded on a single robot, includes only indoor scenes.
OUT_Train	OUT	27,368	Yes	No	Curated subset of *Outside*
K3D_Train	R2X	70,745	Yes	No	*R2X_Train* dataset restricted to field-of-view of K3D robot (see Fig. 2)

frames remained which we distributed into 13,791 (20%) frames for testing and 56,954 (80%) for training.

For *outdoor* obstacle avoidance (robot *OUT*) we collected 66,057 frames in a variety of outdoor settings. These were collected on a mobile phone connected via Wi-fi to an *OUT* robot. The phone also translated the phone-paired Bluetooth controller commands to Wi-fi and sent them to the robot. These steps were necessary since at that time no Compute Module with significant onboard memory was available.[11] The frames were collected in 1280x720 resolution in raw H264 format at 15fps. A preliminary curating step removed those sequences where the robot stood still, was being repaired or when movement commands were not accepted by the robot. This yielded the 51,735 frames for *Outside* in Table 2. We then used the same additional filtering as above and obtained 27,368 frames which we distributed into 5,351 (20%) for testing and 22,017 (80%) for training.

In all cases except *K3D* the frames were downscaled to half recording resolution via linear interpolation while preserving aspect ratio. For *K3D*, where only a part of the frame was used for training and testing due to its much smaller horizontal field-of-view (see Fig. 2), the stereo views were not scaled down but used at the original resolution.

For the qualitative evaluation of stereo algorithms we used the stereo images as-is (except for linear up- and downscaling). For the disparity-sensitive deep learning model (*NCC-Disp*) experiments, we instead computed the normalized correlation coefficient (see Eq. 1) between patches of the **L**eft and **R**ight stereo views using 8x8 pixel-sized patches and disparities of 0, 1, 2, 4, 8, 16 and 32 pixels. Patches were moved in half-patch-size (4 pixel) steps. We obtained seven channels with a resolution of 78x44 pixels for *OUT* and *R2X*, but 62x48 pixels for *K3D* due to the different aspect ratio and resolution.

[11]The old RPi ComputeModule evaluation board offers neither an SD card slot nor Bluetooth and the only available USB slot was needed for a Wi-fi USB stick.

5 QUALITATIVE EVALUATION OF STEREO ALGORITHMS

We evaluated the following stereo algorithms, using the stereo views described in the previous section. We ran the algorithms on our datasets from Table 2. In each case, we chose a random sample of about 50 frames and evaluated them w.r.t. the usefulness for obstacle avoidance into three categories. We also inspected extensive simulated runs of each algorithm where frames were processed in the same order as they would be processed on the robot.

- **Good:** Good stereo depth reconstruction with at most small errors.
- **Mediocre:** Stereo depth reconstruction with major errors but still somewhat useful for obstacle avoidance.
- **Bad:** Incorrect stereo depth reconstruction with at most small patches with approximately correct depth; useless for obstacle avoidance purposes.

Since we were interested in comprehensive performance, we chose the largest, most diverse dataset *Mixed*, and for promising algorithms additionally the dataset *Outside* to enable some comparison w.r.t. natural scene performance.

1. GraphCut (GC, from Kim et al. (2003) as *KZ1*)
2. Block Matcher (BM, Konolige (1998))
3. Semi-Global Block Matcher (SGBM, Hirschmuller (2008))
4. HITNet (Tankovich et al. (2021))

Reflections on the wooden parquet floor in dataset *Mixed* sometimes made the system merge objects with their reflections, making them seemingly appear below the floor. As this must be addressed separately, we have disregarded such errors in our evaluation.

We used the OpenCV (Bradski (2000)) implementation of all algorithms except HITNet, which was available as a GIThub project with pretrained Ten-

$$NCC(i,j,d) = \frac{\sum_{\forall i',j' \in [0,7]} L(4i+i',4j+j')R(4i+i'-d,4j+j')}{\sqrt{\sum_{\forall i',j' \in [0,7]} L(4i+i',4j+j')^2 \sum_{\forall i',j' \in [0,7]} R(4i+i'-d,4j+j')^2}} \quad (1)$$

Table 3: Results of qualitative evaluation. Mixed: 56, Outside: 52 stereo views were randomly selected and manually analyzed by visual inspection. For definition of category labels see text. Last column: proportion of non-bad stereo views.

Stereo Alg.	Dataset	bad	med.	good	Prop. ¬bad
GC	Mixed	48	8	0	14.28%
BM	Mixed	51	5	0	8.92%
SGBM	Mixed	46	10	0	17.85%
SGBM	Outside	39	13	0	25.00%
HITnet	Mixed	25	29	2	55.35%
HITnet	Outside	19	32	1	63.46%

sorflow models.[12] Results can be found in Table 3, however we will also comment on the strengths and weaknesses of the tested algorithms in the following subsections. The latter is based on the simulated runs mentioned earlier.

5.1 GraphCut

GraphCut was one of the first stereo algorithms, however it is also one of the slowest. It is described in Kim et al. (2003) (as *KZ1*). With the default maximum disparity of 16 it is about 100 times slower than the fastest algorithm. Visual inspection shows many large dark areas and black dots where no depth information is reconstructed. During robot turns it becomes completely white indicating minimum distance. A maximum disparity of 40 with two-fold downscaling made it slightly slower but the depth map became very rough and pixelated. Again blobs were often found with other depths that were overlaid over walls or flat objects. Walls were sometimes well detected but only shortly. It does seem as if GC only detected axis-parallel walls and does not deal with gradients. We only ever saw blocks in one color (depth). In a way the best scenes remind of HITNet – except the restriction on axis-parallel walls – but processing speed is of course several orders of magnitudes slower.

5.2 Block-Matching (BM)

Block-Matching (BM, Konolige (1998)) determines disparity by matching blocks between both stereo views. We've again used two-fold linear downscaling for this algorithm. The depth map is very noisy and seems to work only on flat surfaces. Convex surfaces have good disparity values only at the border. The floor shows very little depth information even when structured, e.g. parquet or ceramic tiles. Walls are very hard to discern, and many floor tile seams have completely wrong depth values. In many cases only corners and edges have depth data and the rest of the image is black. When turning, worse depth maps are obtained, and motion blur is clarly discernable – when driving straight ahead for half a second the depth maps stabilizes again and gives better but still not useful results.

[12]https://github.com/google-research/google-research/tree/master/hitnet

5.3 Semi-Global Block Matching (SGBM)

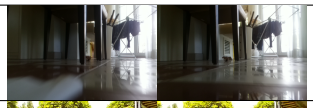
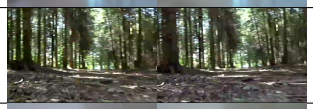
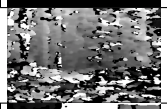
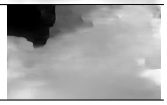
Semi-Global Block Matching (SGBM, Hirschmuller (2008)) is a much more intelligent version of Block-Matching. With linear two-fold downscaling it runs quite fast. When the robot is moving straight ahead walls are often visible relatively well, unlike the floor. When the robot turns, depth information vanishes except for thick edges, but reestablish after a few seconds. Parquet floor and the furniture of the same color is not clearly differentiated, perhaps due its similarity in color and texture. In complex scenes many different patches with different distances are visible. Concluding, SGBM is not perfect but still quite an improvement over BM.

A window size of 9x9 seems to work best and gives reasonable results with simple scenes. For *Outside*, disparities do not change from frame to frame, perhaps because the Compute Module platform had lower latency since both cameras were attached to one module (rather than two modules as for *Mixed*). Trees and skies are sometimes easily discernable but trees are not well separated. Objects are sometimes visible but very noisy and blurry. A smaller size increases noise in estimated depth, a larger size makes disparities change from frame to frame, probably due to the less exact alignment, and makes object contours much more blurry. Robot turns still create artefacts for all tested window sizes of 3x3, 9x9 and 21x21.

5.4 HITNet

The main reason to use HITNet (Tankovich et al. (2021)) was that it was a relatively fast current Deep Learning network that performed well on benchmark datasets, and was also the highest-ranking model on Middlebury (at that time of running the experiments) for which pretrained models were readily available.

Table 4: Sample images for the three categories bad, mediocre and good.

Cat.	Dataset	Stereo views	GC	BM	SGBM	HITnet
good	*Mixed*					
good	*Outside*					
mediocre	*Mixed*					
mediocre	*Outside*					
bad	*Mixed*					
bad	*Outside*					

We evaluated two different pretrained versions of HITNet – eth3D and middlebury – on a random sample of 105 frames from *Mixed* using the pooled version eth3D with two-fold linear downsampling, and middlebury with two-fold linear upsampling.[13] In 48 cases, the depth map was clearly wrong for both versions and we were unable to even approximately match the shown depth maps to the stereo view by visual inspection; in 41 cases, the eth3D version performed better; and in 11 cases, the middlebury version performed better. In 5 cases there was no significant difference. This indicates – not surprisingly – that the eth3D version which was pooled over KITTI, eth3D and middlebury datasets performs much better than the version trained just on Middlebury. About 5 frames looked really well and would have been very useful for obstacle avoidance could this behaviour be obtained everywhere. We therefore chose the pooled eth3D pretrained version for the qualitative evaluation.

HITNet showed small fluctuations even with stereo views that change very little (i.e. when the robot is not moving at all), and strong fluctuations during movement (perhaps due to the random latency of *Mixed* dataset). Every movement – including rotation – shows artefacts. Still HITNet was a significant improvement even over SGBM and sometimes showed good depth maps which even allowed to detect small objects (about every 20th frame). It is also the only algorithm for which we actually found stereo views of category **good**.

[13] According to pers.comm. by V. Tankovich.

Using the *Outside* dataset also indicated that around every second frame was somewhat useful, but very noisy, and also that only very near objects are somewhat reliably detected every second frame, but sadly not in consecutive order – in most cases sequences of bad depth maps alternate with sequences of mediocre depth maps. However in some cases HITNet also hallucinated objects – some of these were due to reflections and were disregarded but others were clear mistakes. The *Outside* dataset does seem to work slightly better than *Mixed*, however this could be due to the better temporal alignment of the stereo views for *Outside*.

It is clear from the results in Table 3 that HITnet is about two to three times better than the best classical algorithm, SGBM, and yields useful (i.e. not **bad**) depth maps about every other frame. It is also clear that stereo views from natural scenes perform better than those from indoor scenes by about 50%. This could be explained by the more complex textures in outdoor scenes or perhaps just the better temporal alignment of stereo views for *Outside*.

As one of the cameras used to record *Mixed* has failed before we could compute rectified stereo, we had to use non-rectified stereo views. To make sure the non-rectified stereo did not influence our results, we also compared HITNet on a random sample of 47 frames from the *Outside* dataset with rectified versus non-rectified stereo views. In 29 cases, the non-rectified version looked better; in 3 cases, the rectified version looked better and in 15 cases there was no significant difference, so in effect the rectified stereo

views performed worse. So we may tentatively conclude that the missing rectification is not responsible for the observed poor performance.

6 DISPARITY-SENSITIVE DEEP LEARNING NETWORK (NCC-DISP)

We designed a series of 25 consecutive networks with different layers and architecture towards the final model presented here. The first one was a straightforward ten-layer model with alternating Conv2D and AvgPool2D layers. The basic principles which in our observation correlate with good performance over this sequence of networks were:

1. **Segmentation:** It proved important to process the slices corresponding to different disparity values independently and only merge them at the last layer. [14]
2. **Dropout:** At a higher level of parameters (more than 500,000) the small dataset led to overfitting. Adding a dropout layer just before the last layer proved to resolve this – and also speeded up convergence – in most cases.
3. **Alternate Convolutional and Averaging Layers:** Alternating convolutional and averaging layers worked reasonably well, and removing averaging layers was harmful to performance.
4. **Use Appropriate Dimensions:** We started out with a 2D representation using seven channels for the different disparity values. However this limited our options for convolutional layers. Switching to a 3D representation of the same data using one channel but seven z-layers enabled using the 3D functions of Conv and AvgPool, which immediately improved the performance.
5. **Use Activation Relu and MaxPool Layers:** Switching from sigmoid to relu activation and replacing AvgPool with MaxPool layers added additional nonlinearity which benefitted overall performance.

[14]The monkey visual cortex and most likely the human visual cortex as well actually splits the different modalities all the way – there is no point where the different levels are connected directly and summed up – so clearly the brain uses at least one other method to merge information without physical connection. One good candidate is clearly neural assembly synchronization, which does not seem to be available in Tensorflow yet.

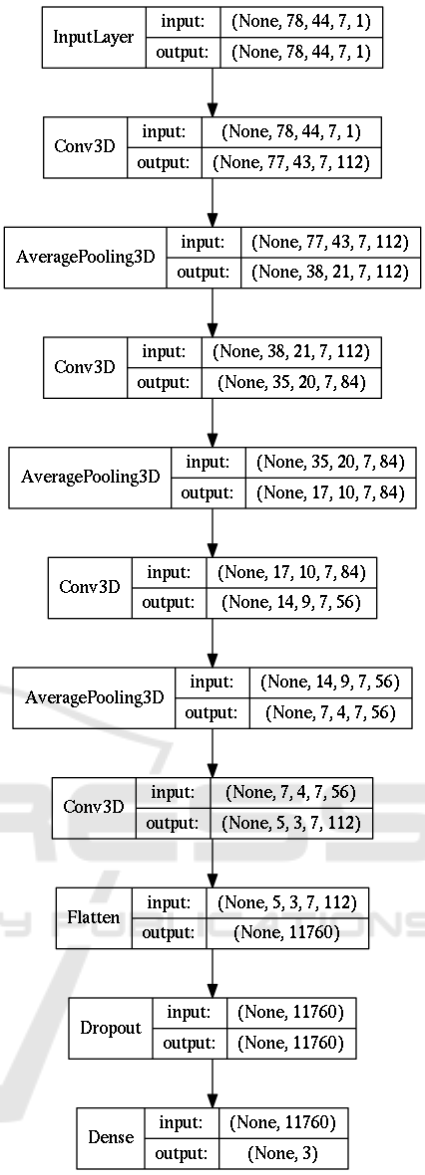


Figure 3: Final disparity-sensitive model NCC-DISP for R2X and OUT. The K3D model was slightly smaller since its input was only [62,48,7,1] but was otherwise the same.

The number of layers was limited by the low resolution of the input dataset, so we could not increase depth significantly without using kernels of very small sizes or removing averaging layers, both of which proved to hamper performance. However as the final model (see Fig. 3) proved similar in evaluation set accuracy to the original model (except for *OUT_Train*), real-life performance is likely to be comparable as well.

Table 5: Results of learning experiments

Dataset	#Samp.	Model	#Param.	#Steps to best model	Corr. coeff.	Eval.Acc.	Acc. w/ center ±48	Acc. w/ center ±32
K3D_Train[15]	70,745	DAVE-like	71,867	278.7k	0.4032	59.58%	59.27%	59.08%
K3D_Train	70,745	NCC-Disp	179,567	285.0k	0.3531	58.65%	58.17%	57.97%
R2X_Train	70,745	DAVE-like	71,867	305.5k	0.3945	59.93%	59.57%	59.05%
R2X_Train	70,745	NCC-Disp	186,623	347.7k	0.4088	59.69%	59.03%	58.80%
OUT_Train	27,368	DAVE-like	71,867	69.5k	0.1807	58.10%	50.27%	40.87%
OUT_Train	27,368	NCC-Disp	186,623	81.8k	0.1349	54.59%	46.01%	39.92%

For training, we used TensorFlow with AdamOptimizer and a learning rate of 10^{-4}. Training data was prepared as described in Sec. 4. We computed normalized correlation coefficient values on regular 8x8 pixel size patches taken from left and right images at seven different horizontal pixel disparity values, yielding a 3D image with a z-Depth seven. Note that due to this transformation no actual image data is fed into the network, essentially forcing it to use disparity information only.

For steering outputs, we formulated steering as a classification problem, representing left, right and straight forward as distinct classes. Left and right were determined from a steering output of ±65 which corresponds to half the maximum steering output of ±127. This variant corresponds to *Cl3* of the original paper Seewald (2020) which was the best-performing output-model there.

Table 5 shows the results of the different models *DAVE-like* (as described in Seewald (2020)) and *NCC-Disp* (see Fig. 3) on the three datasets *K3D_Train*, *R2X_Train* and *OUT_Train*. As can be seen, performance according to evaluation set accuracy and correlation between original and estimated steering is similar between *DAVE-like* and *NCC-Disp* models – except for *OUT_Train* where it is noticably worse. The number of training steps to the best model is also rather similar. The number of parameters for the *NCC-Disp* models is roughly 2.5 times the original *DAVE-like* model but this has been somewhat offset by a dropout layer with 0.8, where 80% of weights are randomly set to zero in each training step. Disregarding the normalized correlation coefficient computation, the model is still well within the size where it can be run on RPi 3B+ or higher.

We were surprised by the performance loss on *OUT_Train*. Initially we assumed that the disparity levels may need to be adapted for the larger robot used to collect this dataset, however experiments with two other disparity sets did not improve performance (data not shown). However, there is one other significant difference: The *OUT_Train* dataset was collected in H264-compressed format while all other datasets were collected in uncompressed YUV 4:2:2, which adds an additional layer of noise on the image data. This could well account for a noisier output from the NCC preprocessing step and therefore for the observed performance loss.

[15]After submitting the final paper version for Seewald (2020), we noticed that by extending the training steps it was possible to enable good performance for this robot as well despite the highly distorted field-of-view. We retrained, took it to the conference and demonstrated it after the session, where it performed reasonably well.

7 DISCUSSION

One limitation of our approach to represent disparity is obviously the simplistic normalized correlation coefficient function to determine patch similarity. It would be better to use a patch similarity computation layer as the first layer of the network and optimize it as a part of the training process. Such a network could adapt receptive field size and disparity sensitivity to actually needed values similar as evolution has done for compound eyes in insects.

One method we have yet not considered is obstacle detection by movement. For a moving robot, simple video analytics such as optical flow can determine objects on collision course by looming. However this method is only applicable once the robot is moving and cannot be used when the robot is standing still. It would perhaps be feasible to extend the deep learning model with recurrent loops to store past frames and integrate such additional information as well. As our data has been collected in long sequences it would be well suited for such learning experiments.

8 CONCLUSION

We have shown that – in the absence of quantitative data – a qualitative evaluation, despite its inevitable subjectivity, is helpful in determining the overall usefulness of algorithms and models for a specific task. In our case of stereo view algorithms we obtained a negative result, however it is clear that deep learning models are improving at a rapid rate and we may expect more progress in the future.

As a second way towards obstacle avoidance with stereo cameras, we have – after noting that deep learning algorithms such as Muller et al. (2004) do not rely on view disparity – proposed the disparity-sensitive deep learning network *NCC-DISP* which has only disparity data as input, and show that it performs as well as an earlier system using raw stereo camera inputs.

The next generation of ToyCollect robots will additionally feature an active sensing depth camera to generate approximate depth maps which would allow a qualitative evaluation of stereo camera algorithms. Adding bumper and acceleration sensors will furthermore allow to automate the generation of large amounts of obstacle detection training data, which is currently still a manual process.

ACKNOWLEDGEMENTS

This project was partially funded by the Austrian Research Promotion Agency (FFG) and by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the Talente internship research program 2014-2019. We would like to thank Lukas D.-B. for all 3D chassis designs of robot *R2X*; Vladimir T. for answering questions and explaining his model; and all interns which have worked on this project, notably Georg W., Julian F. and Miriam T.

REFERENCES

Becker, F. and Ebner, M. (2019). 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,*, pages 167–173. INSTICC, SciTePress.

Bojarski, M., Del Testa, D., Dworakowski, D., et al. (2016). End to end learning for self-driving cars. Technical Report 1604.07316, Cornell University.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. (2016). Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *PAMI*, 30(2):328–341.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., of Biochemistry, D., Jessell, M. B. T., Siegelbaum, S., and Hudspeth, A. (2000). *Principles of neural science, 4th Edition*. McGraw-Hill New York.

Khan, M. and Parker, G. (2019). 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)*, pages 403–411. INSTICC, SciTePress.

Kim, J., Kolmogorov, V., and Zabih, R. (2003). Visual correspondence using energy minimization and mutual information. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1033, Washington, DC, USA. IEEE Computer Society.

Konolige, K. (1998). Small vision systems: Hardware and implementation. In *Robotics research*, pages 203–212. Springer.

Luo, W., Xiao, Z., Ebel, H., and Eberhard, P. (2019). 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,*, pages 268–275. INSTICC, SciTePress.

Muller, U., Ben, J., Cosatto, E., Fleep, B., and LeCun, Y. (2004). Autonomous off-road vehicle control using end-to-end learning. Technical report, DARPA-IPTO, Arlington, Virginia, USA. ARPA Order Q458, Program 3D10, DARPA/CMO Contract #MDA972-03-C-0111, V1.2, 2004/07/30.

Muller, U., Ben, J., Cosatto, E., Fleep, B., and LeCun, Y. (2006). Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746.

Pfeiffer, M., Schaeuble, M., Nieto, J. I., Siegwart, R., and Cadena, C. (2016). From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. *CoRR*, abs/1609.07910.

Seewald, A. K. (2020). 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.*, pages 652–659.

Tankovich, V., Hane, C., Zhang, Y., Kowdle, A., Fanello, S., and Bouaziz, S. (2021). Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14362–14372.

Wang, Y., Dongfang, L., Jeon, H., Chu, Z., and Matson, ET. (2019). End-to-end learning approach for autonomous driving: A convolutional neural network model. In Rocha, A., Steels, L., and van den Herik, J., editors, *Proc. of ICAART 2019*, volume 2, pages 833–839.