

Deep Obstacle Avoidance*

Keith Sullivan and Wallace Lawson

Naval Research Laboratory, Washington DC, USA

{keith.sullivan, ed.lawson}@nrl.navy.mil

Abstract

We present work on a robot system capable of rapidly learning to avoid obstacles in previously unseen environments. We exploit deep learning’s generalized features to reason about when to turn left, turn right, or drive straight within an indoor environment. We present preliminary results of a deep convolutional neural network trained without any pre-training that can successfully avoid obstacles.

1 Introduction

For robots to be truly autonomous, they must be able to successfully navigate through their environment, which inherently means avoiding any obstacles that are present. Typically, robots accomplish obstacle avoidance using a ranging sensor, and many algorithms have been developed assuming the presence of a ranging sensor. Typical ranging sensors include sonar, laser, optical flow, and depth sensors. However, all these sensors suffer from various issues: sonar is susceptible to multi-path reflections and provides sparse angular resolution; lasers provide denser angular resolution but are complex and expensive; optical flow is computationally expensive; and depth sensors tend not to work outdoors.

Ranging sensors provide a 2D slice of the environment, which can cause problems for a 3D robot. While 3D laser scanners can solve this problem, they produce large amounts of data that is difficult to process quickly. For example, the highly popular Velodyne HDL-64E produces over 2,200,000 3D points per second.

Replacing a ranging sensor with a camera allows a fuller understanding of the environment which leads to better obstacle avoidance. Cameras operate in a wide variety of lighting conditions, are lightweight and low-powered enough to allow deployment on a variety different robotics platforms such as ground and aerial vehicles. Still, processing image data quickly enough to allow reasonable robot translation speeds is a challenge.

Deep learning has shown great promise across a wide variety of image based problems. The primary advantage of

using deep learning for obstacle avoidance is to remove the difficulty in a priori knowing the appropriate features for the current operating environment. While hand-crafted features tend to work well for obstacle avoidance, they typically fail when the robot is placed in environments that differ from the training environment. Deep learning alleviates this issue by using all the pixel information and does not rely on tuning camera parameters, geometry, or camera calibration. Additionally, with current GPU technology, deep learning can run at frame rate, sufficiently fast enough for accurate robot control.

This paper presents a robot capable of performing obstacle avoidance in an indoor environment, where the system is capable of rapidly learning features to avoid obstacles. By exploiting deep learning’s general image features, we can quickly train a robot to avoid obstacles in a previously unseen environment without need to make limiting assumptions about environment including texture, illumination, and geometry. In this paper, we train the robot to navigate through the environment by providing labeled examples of situations where it should turn left, turn right, or go forward. This information is used to train to a CNN with output neurons that translate directly to navigation commands. We demonstrate the performance of the algorithm in successfully navigating through an environment.

2 Related Work

Vision-based techniques for obstacle avoidance fall into two broad categories. The first category attempts to compute the apparent motion – optical flow – of objects within the scene, and based on this estimated motion, determine a control input to avoid obstacles [Carloni *et al.*, 2013; Herisse *et al.*, 2008; McCarthy and Bames, 2004]. Optical flow works well in environments with enough texture to track individual pixels across multiple frames, but fails in broad texture-less areas or when illumination changes [Guzel and Bicker, 2010]. Additionally, optical flow is computational expensive without specialized hardware [Günyel *et al.*, 2012].

The second category uses pixel-level texture information to determine the appropriate control to avoid obstacles [Bonin-Font *et al.*, 2008; Desouza and Kak, 2002]. Obstacles are defined as pixels that differ significantly from the ground and are detected using standard computer vision features such as SIFT [Guzel and Bicker, 2011] and FAST [Saitoh *et al.*,

*Keith Sullivan was funded by the Naval Research Laboratory under a Karles Fellowship, and Wallace Lawson was supported by the Office of Naval Research.

2009]. These techniques fail in environments with visually similar regions, wide illumination differences, and different terrain types with geometrically similar structures.

Closest to our work, Giusti et al developed a CNN to control a quadcopter flying in a forest [Giusti *et al.*, 2016]. The authors collected training data by having a human walk with a camera through the woods, and then trained a CNN to produce control commands to control the quadcopter for following a path through the forest. While their work is directed more towards following a specific path, we focus on general navigation.

3 Network Structure

We use a convolutional neural network (CNN) to control the robot. The network takes a RGB image as input. The CNN has three outputs: turn left, turn right, and go forward. The robot implements these outputs as constant rotational and/or translational velocities.

The CNN architecture consists of alternating convolutional layers [LeCun *et al.*, 2012] with max pooling layers [Scherer *et al.*, 2010], followed by two fully connected layers. The convolutional and pooling layers extract geometric information about the environment while the fully connected layers as a general classifier. In particular, our CNN has the following structure:

- *conv1* → Convolution layer of 75 filters of size 10x10 with stride of 10
- *pool1* → Max pooling layer with kernel of 3x3 and stride of 2
- *conv2* → Convolution layer of 75 filters of size 5x5 with stride of 1
- *pool2* → Max pooling layer with kernel of 2x2 and stride of 2
- *conv3* → Convolution layer of 75 filters of size 5x5 with stride of 1
- *pool3* → Max pooling layer with kernel of 2x2 and stride of 2
- *fc1* → Fully connected layer of size 500
- *relu1* → Rectified Linear Unit
- *fc2* → Fully connected layer of size 500
- *relu2* → Rectified Linear Unit
- *softmax* → SoftMax layer with 3 outputs

The convolutional layers perform 2D convolution of their input maps with a rectangular filter. When the previous layer contains more than one map, the results of the convolutions are summed and transformed by a hyperbolic tangent activation function. Higher activations occur where the filter better matches the content of the map. The output of the max pooling layers is the maximum activation of non-overlapping square regions of the input, and these layers simply select the winning neurons. The final softmax layer computes the activation for each of the three classes, which can be interpreted as the probability of the input image requiring that robot control.

	Without Obstacle	With Obstacle
Forward	2133	648
Turn Left	2054	308
Turn Right	2319	0
Total	6506	956

Table 1: Number of training images

We implemented our CNN using Caffe [Jia *et al.*, 2014], and trained the network without any pre-training. The weights were randomly initialized, and then adaptive gradient descent [Duchi *et al.*, 2011] was used to minimize the loss over the training set. We choose adaptive gradient descent to maximize the predictive value of highly predictive but seldom seen features.

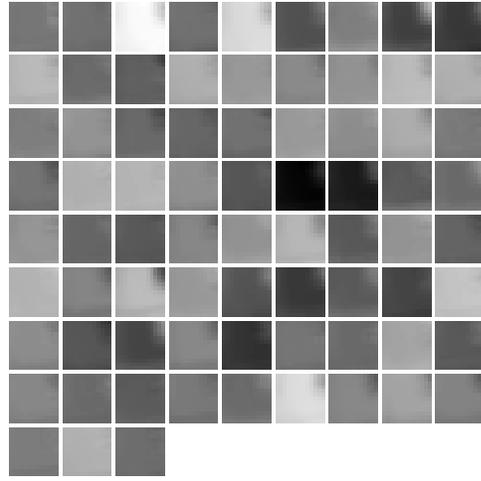
Figure 1 shows a visualization of the 75 kernels of the *conv3* layer for representative images for turn left, turn right, and go forward. More white regions indicate a positive activation, darker regions indicate a negative activation, and grey regions indicate no activation. Note that even with our (relatively) small training set, the CNN has learned different features for each output from the softmax layer. There are some interesting differences in the responses from the *conv3* layer. For example, many of the features from turning and quite similar to each other. At the same time, the differences between features for turning (left or right) and going forward are quite different. It’s clear that the network has been able to key in on some distinctive features of the environment, and that these features are visible in cases when the robot should move forward.

4 Demonstration

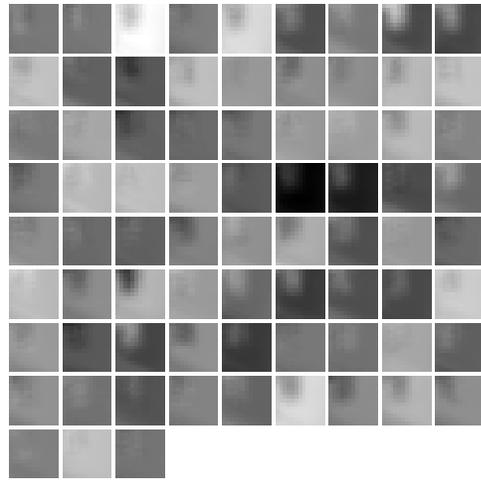
To demonstrate our system, we constructed a constrained environment for a ground robot. Figure 2 shows the environment which measured approximately 4.8 meters by 4.8 meters with walls approximately 1.25 meters high. We used a Pioneer 3AT robot equipped with a Carnegie Robotics SL camera/laser system, although we did not use the laser for these experiments. The camera produced images at 1024x1024 resolution, and due to the circular projection, we cropped out the center 750x750 for input into the CNN. Figure 3 shows the complete robot. For training, we tele-operated the robot around the environment to collect labelled images, while during testing, the robot was fully autonomous with all calculations occurring on the on-board laptop running a NVIDIA GeForce GTX 980M graphics card.

Training data was collected both without an obstacle (a black plastic box visible in Figure 2) and with the obstacle present. Table 1 shows the total number of training images along with the breakdown between classes. For this paper, we assume the robot will only turn left to avoid the obstacle.

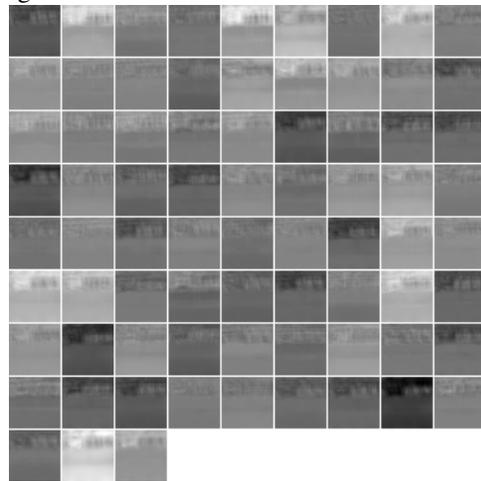
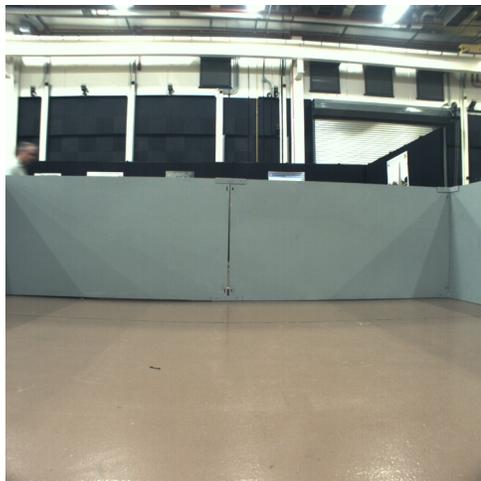
Overall it took about 30 minutes to collect all the training images without the obstacle present, and an additional 30 minutes to train the CNN without any pre-training. At this point, the robot successfully avoided the walls, and avoided the obstacle independent of its location within the playpen.



Turn Left



Turn Right



Forward

Figure 1: Visualization of the features from the *conv3* layer for example images for turn left, turn right, and go forward.

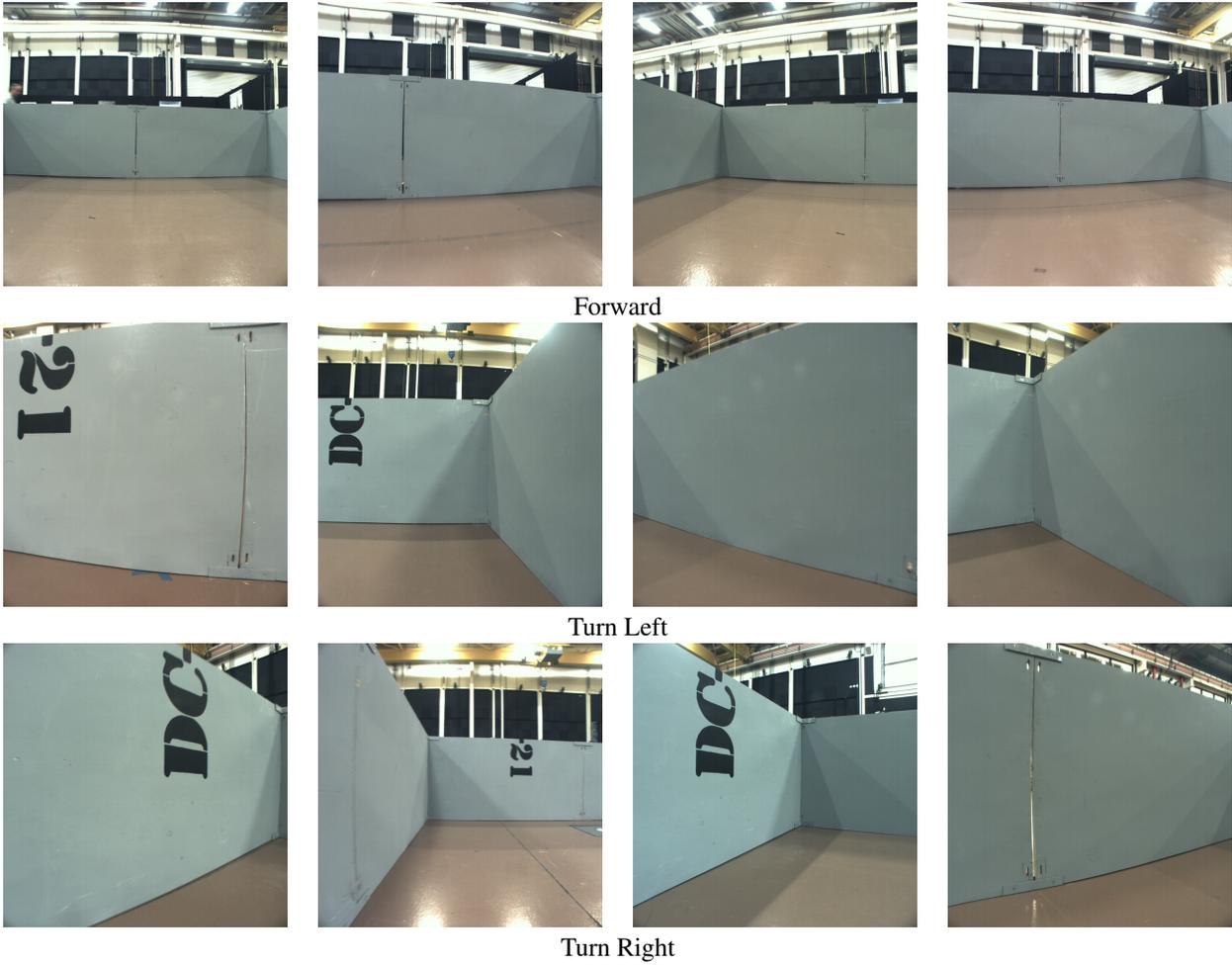


Figure 4: Example training images. The top row is go forward, the middle row is turn left, and the bottom row is turn right.

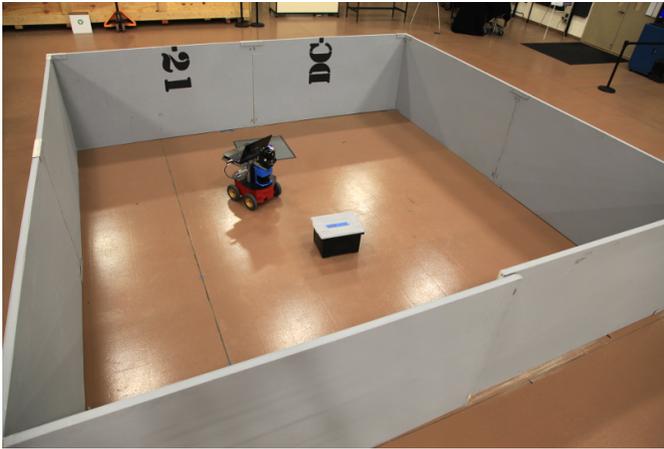


Figure 2: Robot playpen



Figure 3: Our robot

5 Discussion

Our obstacle avoidance approach works in simple indoor environment, but does demonstrate the deployment of a supervised machine learning technique to a robotics problem. There are several aspects of the problems worth investigating in future works. First is the issue of saliency in the image. From the perspective of the ground robot, for example, it may not be necessary to look at pixels that are high in the image, since those have no bearing its current navigation. This may not a problem in the constrained environment, but it could be an issue in other situations, such as when there may be low hanging obstacles.

Another interesting aspect for investigation is shifting from supervised to semi-supervised or perhaps completely unsupervised learning. A challenge deploying supervised learning to mobile robots is collecting labeled training data. The technique we used in this paper – teleoperation followed by manual labeling – works, but is time consuming, error prone, and does not scale to larger action spaces.

As a first step to solve these issues, we plan to develop a teleoperation system that will collect labeled images in real-time as the human drives the robot: the labels will be based on joystick positions. Similar to learning from demonstration (LfD) [Sullivan *et al.*, 2010; Sullivan, 2015], the human will drive the robot around the environment collecting sufficient training data, and then a CNN will be constructed (e.g., a policy in LfD parlance). The demonstrator can then observe the robot autonomous driving through the environment, and collect additional training images to correct any errors.

The ultimate goal for our system is to have the robot learn independent from human interaction. In other words, we would the robot to learn to avoid obstacles without human intervention. We foresee the system working as follows: the robot would initially have a CNN with random weights and would use this random CNN to move within the environment (the robot would have some reinforcement signal to encourage forward movement rather than simply spinning in place). As the robot moves around, it records images and labels them with the output of the CNN. Once the robot senses an imminent crash (via a laser, say) it would then enter training mode to update its CNN. After updating the CNN, the procedure begins anew, but with a now (hopefully) better performing CNN. In this manner, we should be able to bootstrap obstacle avoidance behavior within any environment. This approach helps to move deep learning from a fully supervised approach to a semi-supervised approach, which is more amenable to a robotics implementation. Some recent work has used this approach to teach a robot to grasp arbitrary objects [Lenz *et al.*, 2015], but, to date, no research has sought to use semi-supervised deep learning for robot control.

Finally, recent advances in neuromorphic technologies have generated low-power extremely high frame-rate enabling technologies that can be used in an application such as deep obstacle avoidance. While the use of a laptop with a GPU on it can work for a limited amount of time, it still consumes a significant amount of power, and cannot operate for long periods of time without needing to be re-charged. Further, in applications where a robot is moving quickly through

a cluttered environment, responses that are in the millisecond range may be needed to avoid obstacles.

References

- [Bonin-Font *et al.*, 2008] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.
- [Carloni *et al.*, 2013] Raffaella Carloni, Vincenzo Lippiello, Mario D’Auria, Matteo Fumagalli, Abeje Y Mersha, Stefano Stramigioli, and Bruno Siciliano. Robot vision: obstacle-avoidance techniques for unmanned aerial vehicles. *Robotics & Automation Magazine*, 20(4):22–31, 2013.
- [Desouza and Kak, 2002] G. N. Desouza and A. C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, Feb 2002.
- [Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July 2011.
- [Giusti *et al.*, 2016] Alessandro Giusti, Jerome Guzzi, Dan Ciresan, Fang-Lin He, Juan Pablo Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jurgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016.
- [Günyel *et al.*, 2012] Bertan Günyel, Rodrigo Benenson, Radu Timofte, and Luc Van Gool. Stixels motion estimation without optical flow computation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 528–539. Springer, 2012.
- [Guzel and Bicker, 2010] Mehmet Serdar Guzel and Robert Bicker. Optical flow based system design for mobile robots. In *Proceedings of Robotics Automation and Mechatronics Conference (RAM)*, pages 545–550. IEEE, 2010.
- [Guzel and Bicker, 2011] Mehmet Serdar Guzel and Robert Bicker. *Vision based obstacle avoidance techniques*. INTECH Open Access Publisher, 2011.
- [Herisse *et al.*, 2008] Bruno Herisse, Francois-Xavier Rusotto, Tarek Hamel, and Robert Mahony. Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 801–806. IEEE, 2008.
- [Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, pages 675–678, New York, NY, USA, 2014. ACM.
- [LeCun *et al.*, 2012] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Neural Networks: Tricks of the Trade: Second Edition*, chapter Efficient BackProp, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [Lenz *et al.*, 2015] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.
- [McCarthy and Bames, 2004] Chris McCarthy and Nick Bames. Performance of optical flow techniques for indoor navigation with a mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 5093–5098. IEEE, 2004.
- [Saitoh *et al.*, 2009] T. Saitoh, N. Tada, and R. Konishi. Indoor Mobile Robot Navigation by Central Following Based on Monocular Vision. *IEEJ Transactions on Electronics, Information and Systems*, 129:1576–1584, 2009.
- [Scherer *et al.*, 2010] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III, ICANN’10*, pages 92–101, Berlin, Heidelberg, 2010. Springer-Verlag.
- [Sullivan *et al.*, 2010] Keith Sullivan, Sean Luke, and Vittoria Amos Ziparo. Hierarchical learning from demonstration on humanoid robots. *Proceedings of Humanoid Robots Learning from Human Interaction Workshop*, 38, 2010.
- [Sullivan, 2015] Keith Sullivan. *Hierarchical Multiagent Learning from Demonstration*. PhD thesis, George Mason University, 2015.