2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

A Deep-Network Solution Towards Model-less Obstacle Avoidance

Lei Tai[1], Shaohua Li[2], Ming Liu[1,2]

Abstract— **Obstacle avoidance is the core problem for mobile robots. Its objective is to allow mobile robots to explore an unknown environment without colliding into other objects. It is the basis for various tasks, e.g. surveillance and rescue, etc. Previous approaches mainly focused on geometric models (such as constructing local cost-maps) which could be regarded as low-level intelligence without any cognitive process. Recently, deep learning has made great breakthroughs in computer vision, especially for recognition and cognitive tasks. It takes advantage of the hierarchical models inspired by human brain structures. However, it is a fact that deep learning, up till now, has seldom been used for controlling and decision making.**

Inspired by the advantages of deep learning, we take indoor obstacle avoidance as example to show the effectiveness of a hierarchical structure that fuses a convolutional neural network (CNN) with a decision process. It is a highly compact network structure that takes raw depth images as input, and generates control commands as network output, by which a model-less obstacle avoidance behavior is achieved. We test our approach in real-world indoor environments. The new findings and results are reported at the end of the paper.

I. Introduction

A. Motivation and bio-inspired perception

Since the last decade, deep-network structures have been adopted ubiquitously not only in robotics [1], [2], [3], [4], but also natural language processing [5], [6], [7], [8], computer vision [9] and so on. When distinct input and output state spaces can be defined, the related approaches using complex hierarchical structures are generally known as *deep-learning*. Deep-learning is a typical bio-inspired technology. It origins from the artificial neural network (ANN) paradigm [10], which was invented in 1940s by McCulloch and Pitts. ANN tries to simulate the nervous system, where the information is preserved and transmitted through a network structure. From the perspective of roboticists, deep-learning should ultimately mimic a human brain and solve the perception and decision-making problems, which has not been fully developed yet. However, deep-learning has successfully, at least in part, solved several preliminary perception issues for robots, just like what a human brain can do. [11], [9] solved the visual perception as object recognition. [12] solved the acoustic perception. Regarding the decision-making, a recent work by Google DeepMind has shown that the decision process could be learned using a deep reinforcement learning model on the *Q-functions* [13], [14]. Note that all these state-of-the-art methods tried to solve only one aspect of perception, or with strong assumptions on the distinctiveness of the input features (e.g. pixels from a game-play). However, a practical robotic application is usually conducted with uncertainties in observations. This requirement makes it hard to design an effective and unified deep-network that is able to realize a complete - though maybe seemingly simple - robotic task. Despite these considerations, we present in this paper a unified deep-network that is able to perform efficient and model-less robotic exploration in real-time.

B. The need of deep-learning in robotics

Compared with other research fields, robotics research has particular requirements, such as we need to take into account the uncertainty in perception and the demand of real-time operation. Robotics research is generally task-driven, instead of precision-driven. A prestigious computer vision recognition algorithm may result in almost-perfect precision. However, the tiniest undetectable flaw may result in failure of a complete robotic mission. Therefore, the balance among the real-time capability, precision and confidence of judgment is specifically required in robotics. Although there are several libraries for deep-learning in computer vision and speech recognition, we are still in need an ultra-fast and reliable library for robotic applications. We have recently presented a novel deep-learning library - *libcnn* [1], which is optimized for robotics in terms of lightweight and flexibility. It is used to support the implementation of all the related modules in this paper.

C. Contributions

The core contribution of this work is that we present a deep-network solution towards model-less obstacle avoidance for a mobile robot. It results in high similarity between the robotic and human decisions under the same situations, which partially leads to effective and efficient robotic exploration. This is the first work to en-couple both robotic perception and control in a real environment with a single complex network.

D. Organization

The remainder of the paper is organized as follows: Section II will go through the recent related works in deep-learning as well as decision-making approaches, followed by a brief introduction to the proposed deep-network model in Section III. After that, we will introduce the validation experiments of the proposed model in Section IV. We discuss

*This work was supported by the Research Grant Council of Hong Kong SAR Government, China, under project No. 16206014 and No. 16212815; National Natural Science Foundation of China No. 6140021318, awarded to Prof. Ming Liu

[1]Lei Tai and Ming Liu are with MBE, City University of Hong Kong. lei.tai@my.cityu.edu.hk

[2]Shaohua Li and Ming Liu are with ECE, Hong Kong University of Science and Technology. slias,eelium@ust.hk

[1]http://github.com/libcnn

the pros-and-cons and potential use-cases of the proposed model in Section V. At the end, we conclusion the paper and provide additional reference to related materials.

II. Related work

The deep-network usually functions as a standalone component to the system nowadays. For example, D. Maturana *et. al* proposed an autonomous UAV landing system, where deep-learning is only used to classify the terrain [2]. Additionally, deep-learning has also been used to model scene labels for each pixel, as described in [15], [16], leading to semantic mapping results.

A. Deep learning for computer vision

In order to address the previously mentioned problems and challenges, a variety of deep neural networks were reported in the literature. Most of these solutions took advantage of the convolutional neural network (CNN) for feature extraction. In our previous work [17], motivated by the need of real-time computing for robotic tasks, we propose a PCA-based CNN model to remove data redundancy in a hidden layers, which ensured the fast execution of a deep neural network. Besides these models, a number of regularization algorithms have been proposed. In 2014, a fully convolutional neural network was proposed by J. Long *et. al* [16] that highly reduces the computation redundancy and could be adapted to inputs of an arbitrary size. In this work, the concept of deep-network is further extended from perception to decision-making.

Although these proposed approaches have been validated on typical data-sets, it is still questionable how well these methods would perform considering practical conditions. Besides, the task of recognition could only be considered as an intermediate result considering robotic applications, and further reasoning and decision making is required. Meanwhile, one-stroke training strategy, as widely used by computer vision researchers, may not be suitable considering robotic applications. It is more suitable to use reinforcement learning algorithms to allow the system improve performance and increase confidence after each decision made.

B. Deep learning for decision making

Recently, Q-learning models have been adapted to deep neural networks [13]. V. Mnih *et al.* [14] successfully utilized CNN with Q-learning for human-level control. The proposed approach has been validated on several famous games. Results show that the proposed system performs well when dealing with problems with simple states. While when it comes to problems that requires much reasoning, the performance of the system gets poorer. Besides, since the input is the screen of a game, probability and uncertainty were not considered in that model. Tani *et al.* proposed a model-based learning algorithm for planning, which was using a 2D laser range finder and was validated with simulation.

Although the above mentioned models put deep neural networks into applications of decision making, and Q-learning strategy is introduced in the learning process, it was still not convincing how well deep-learning could help real world applications. The problem of game playing occurs in a simulated environment. When it comes to real world applications, a lot more factors should be taken into consideration, like the definition and description of states, the introduction of noise from real data, etc. In this work, we directly take the raw sensor measurements as input the the network, and the robot control commands are directly generated from the hierarchical network without modeling the local environment.

III. A Hierarchical Network for Obstacle Avoidance

In this section, we are going to give a brief introduction to the proposed model, which is used to generate control command for exploration of an indoor environment. It comprises a CNN front-end network for perception and a fully-connected network for decision making. The CNN generate a set of high-dimensional feature maps, which are taken as input to the fully-connected layers. All in all, these two networks seamlessly join together. Such a complex network takes raw sensor data as input and robot control commands as output directly. The overall structure is shown as figure 1.

A. CNN for perception

Convolutional Neural Network (CNN) is one type of hierarchical neural networks for feature extraction. By back-propagating the gradients of errors, the framework allows to learn a multi-stage feature hierarchy. Each stage of the feature hierarchy is composed of three operations: convolution, non-linear activation and pooling.

1) Convolution: The convolution operation does the same as image filtering. It takes the weighted sum of pixel values in a receptive field. It has been proved that a larger receptive field would contribute to the classification error. The mathematical expression of convolution is denoted as follows:

$$y_{ijk} = (W_i * x)_{jk} + b_i \tag{1}$$

where, y_{ijk} denotes the pixel value at coordinate (j, k) of the i-th output feature map. W_i denotes the i-th convolution kernel, x is the input and b_i is the i-th element of the bias vector, which corresponds to the i-th convolution kernel.

2) Non-linear activation: After convolution, an element-wise non-linear function is applied to the output feature maps. This is inspired by the biological nerve system to imitate the process of stimuli transmitted by neurons. the sigmoid function $s(x) = \frac{1}{1+e^{-x}}$ and the hyperbolic tangent function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ was firstly used for activation. Later a piece-wise linear function, namely rectifier, is widely used, which is defined as follows,

$$f(x) = \max(0, x) \tag{2}$$

A neuron employing the rectifier is also called a rectified linear unit (ReLU). Due to its piece-wise linear property, the rectifier executes faster than the previous two non-linear functions for activation.

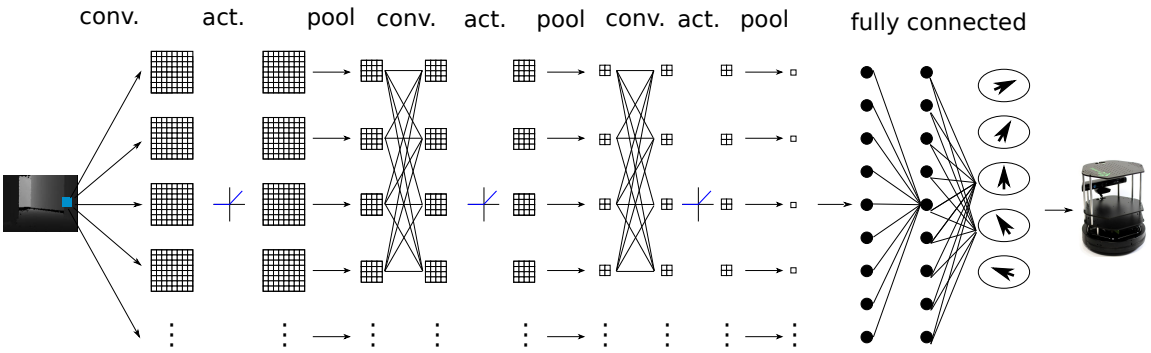


Fig. 1. The proposed model which combines CNN with fully-connected neural network for robot control. Note that conceptually, the CNN section mimics the function of visual cortex and the fully connected layers mimic the premotor cortex.

3) Pooling: The pooling operation takes the maximum or average value (or a random element for stochastic pooling) in an image patch. Pooling aims to improve the robustness of a network and reduces the effect of noise observations.

4) Stride: The stride parameter exists in the convolution layer as well as pooling layer. It means the step over pixels of convolution by patch-by-patch scanning. When stride $s > 1$, the output feature maps is down-sampled by a factor of s. By introducing the stride parameter, the parameter size of the whole network is reduced.

B. Supervised Confidence-based Decision Making

Per the proposed structure shown in figure 1, we take depth images as the only input of the network. The depth maps provide straightforward information of traversability. Unlike traditional computer vision applications, where each label of the output represents either an object or scene categories, the output of our model are control commands. We consider the control commands are regression results from a fully-connected network. To make a decision and generate control commands, we propose the following approach:

1) Control command discretization

Generally, the output commands can be generated by a linear classifier. For this multi-label case, a soft-max classier is used in our model. To achieve the control of the robot, the output control commands are sampled and discretized. Considering the output state space, both the linear and rotational speeds are analog. The greater number of discretization levels leads to the higher precision in control, however with greater computational complexity and more difficulties in network convergence, *vice versa*. We empirically choose five discretization steps for the control command, which are "turning-full-right (0)", "turning-half-right (1)", "go-straightforward (2)", "turning-half-left (3)" and "turning-full-left (4)". In other words, a set of pre-set rotational velocities are defined as a discretized angular velocity space, i.e. $\vec{\Omega} = (\omega_0^*, \omega_1^*, \omega_2^*(=0), \omega_3^*, \omega_4^*)^T$, where ω_i are parameters for discretized control.

2) Confidence-based regression

The fully-connected layers work as a predictor of co-efficients. These coefficients server the linear combination of feature maps generated by the CNN. On this regards, we adopt a confidence-based decision making strategy. The output of the soft-max classifier are taken as the probability of each label (discretized directions). Note that it solves the shortcomings of a winner-take-all strategy. For example, if the highest possibility is 0.3 ()the agent is supposed to take a right turn), while the second highest possibility is 0.29 (the agent to go straight forward). According to the first strategy, the agent turns right. While the fact is that the agent is not sure whether to turn right or go straight forward. This could be solved by the linear combination of the decision candidate. Let c_1, c_2, c_3, c_4, c_5 denote the confidence of each output label, and ω_a denote the angular velocity of the of the mobile robot. The output angular velocity is

$$\omega_a = < \vec{\Omega}^*, (c_1 \ c_2 \ c_3 \ c_4 \ c_5)^T > \tag{3}$$

where $< \cdot, \cdot >$ is an operator of inner-product. Equation (3) maintains a trade-off among different output decisions. Note that it is seemingly likely that the initial decisions can be wrongly averaged by coincidentally simultaneous left and right decisions. However, these dilemma cases do not present in real experiments, considering the feature maps are largely different for these opposite cases.

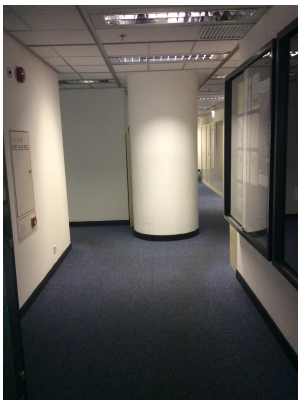
IV. Experiments and Results

A. Platform and environment

In order to validate the effectiveness of our proposed model, we use a TurtleBot for experiments. To acquire visual inputs, a Microsoft Kinect sensor is equipped, for which the effective sensing range is 800 mm to 4000 mm. We use the open source framework Robot Operating System(ROS)[2] to integrate the test system. All the codes are running on a low performance laptop with an Intel Celeron processor without GPU. Based on our recent results of PCA-based CNN, it performs real-time execution of the deep neural network [17]. The system is shown in Fig. 3(a). The test environment we used is an indoor environment with corridors. A typical scene is shown Fig. 3(b).

[2]http://www.ros.org

(a) Hardware and platform



(b) Sample test environment

Fig. 2. Platform and sample test environment

B. Data Gathering

In our experiment, we use a set of indoor depth data-sets for training. The ground-truth output is instructed by a human operator. To be more specific, during the training process, an instructor operates the mobile robot to explore an unknown indoor environment without colliding into obstacles. The robot with the proposed complex network will learn these experience, and adapts these experience in new environments. We record the synchronized depth maps by Kinect and the control commands by the human operator. This dataset is used to train the proposed model in Section III. Note that the the control commands are sampled and discretized into five categories corresponding to the discretized control labels.

C. Network Configuration

The original depth image size from Kinect is 640×480. In our experiment, the input size is down-sampled to a quarter of the original size, i.e. 160×120. This operation will largely reduce the computational cost without introducing much errors. The down-sampled depth map is put into a three-stage "convolution + activation + pooling" cycles, followed by one fully connected layer for feature extraction. The first convolution layer uses 32 convolution kernels of size 5×5, followed by a ReLu layer and a 2×2 pooling layer with stride 2. The second stage of convolution + activation + pooling is the same as the first stage. For the third stage, 64 convolutional kernels of size 5×5 are used, with no change of the ReLu layer and pooling layer. This results in 64 feature maps of size 20×15. The fully-connected layer consists of 5 nodes. The last layer represents the score of each output state. The control commands consist of 5 states: one for going straightforward, two for turning left and two for turning right as defined previously. The final decision is calculated by applying regression using the outcome of the last layer as co-efficiencies over the 5 possible control commands.

D. Sample Results and Evaluation

In all trails the robot does not collide with obstacles. However, we consider this does not reflect the true performance of the system sufficiently. To evaluate the performance of the system, we study the similarity of the robot decision and human decision under the same situation.

Firstly, we show the results using a soft-max classifier for decision making. We sampled 1104 depth images from the complete indoor data-set where the five categories of control commands are almost equally distributed after selection. This is for the sake of fair comparison over the various cases. Thereby, we use 750 images for training and 354 images for testing. For detail of the data-set, please refer to the last section.

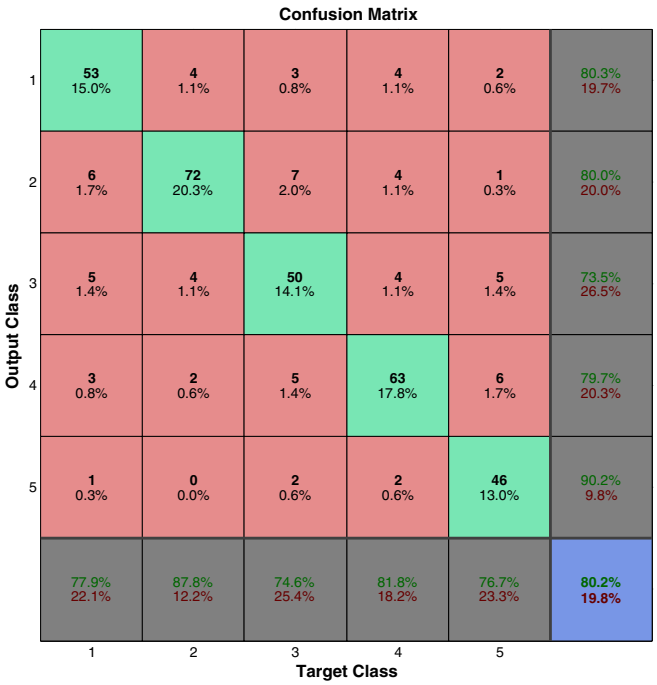


Fig. 3. Confusion matrix on the test set. The green-to-red color-map indicates the accuracy of inference. Note that the outcome is equivalent to a five-labeled classification problem. The result demonstrates outstanding performance of the proposed structure as well as the *libcnn* implementation.

The result is shown in Fig. 3. We could see that the overall accuracy of the test set is 80.2%. The class accuracy is 79.76%, i.e. the mean accuracy of each class. Furthermore, regarding mis-classification, there is quite low chance for our system to generate totally opposite decision, e.g. to mis-classify "left" as "right". A large portion of mis-classifications could be mis-classifying a "turn-half-left" to a "turn-full-left" or "go-straightforward". This further proves the effectiveness of the confidence model, in terms of the error distributions. This result indicates the effectiveness of the system in generating similar controls as human under the same situateions.

Fig. 4 depicts the typical comparison between the decisions made by human and robot over a time series. In this figure, we plot the angular velocity over time, where positive values means turning left and negative means turning right. We set linear velocity as constant. Two series are shown regarding different complexities of the test environment. We sampled 500 points of both the human decision and the

robot decision. We calculated the mean absolute difference between the two cases. In the first case, the mean absolute difference is 0.1114 rad/s. In the second case, the value is 0.1408 rad/s. These statistics indicate that the robot is able to highly imitate the decision from human. Furthermore, the shift in time of making a turning decision is largely due to the sensitive range of the Kinect sensor, making the robot only able to make a turning decision closely in front of an obstacle, while human beings are able to foresee this.

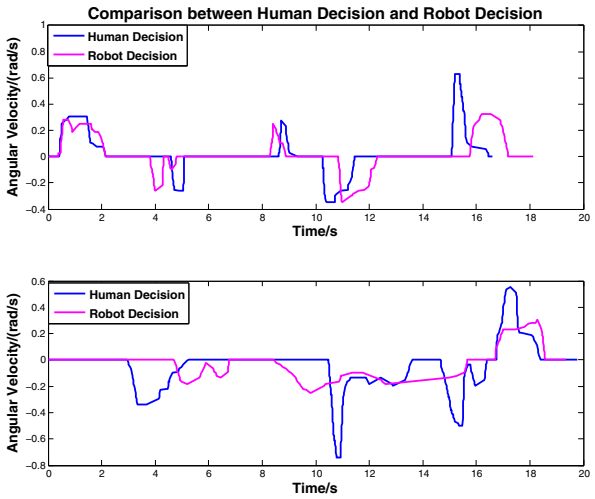


Fig. 4. Comparison of human decision and robot decision

Regarding the running time, the average running time from receiving the input to generate the output command is 247ms, with a variance of 12ms. Note that we get this real-time performance without using any GPU to attempt faster evaluation of the complex network.

V. Discussion and Analysis

In our model, a complex network is used to enable a mobile robot to learn a feature extraction strategy. Although the model is trained in a supervised way, the use of CNN avoids the calculation of hand-crafted features, which allows the agent better adapt to different kinds of environments. Besides, the hierarchical structure of the network maps the input to a higher dimension and enables the successful application of a linear classifier.

A. Visualization of feature maps

In order to demonstrate how the trained network functions as reasonable visual perception functions, we visualize the feature maps generated from the last layer of the CNN in Fig. 1. The second-last layer represents the feature maps, which are further categorized by the fully connected layer. The outcome leads to the selection of control commands. However, it is not easy to visualize the feature maps which are denoted by

$$G := \{g_i | g_i = NN(I_i)\}$$

where NN is the function of the deep-network and I_i is the input depth image with index i. g_i is the corresponding feature map of the input i. Note that each g_i is with dimension of $g_i \in \mathbf{R}^{19200}$. To solve the visualization problem, we adopt a latent variable Gaussian Process model (GPLVM), more specifically, with Spike and Slab Gaussian Process Latent Variable Models (SSGPLVM), which was recently proposed by Dai et al. [18]. We project g_i into a lower dimensional space [3]. After that, a pair of distinguished dimensions can be visualized as $\mathbf{R}^2$. The final visualization result is shown as Fig. 5. The legend indicates the label of the generated control commands, i.e. 0 refers to that the robot will make a full-right turn; 4 indicates a full-left turn and 2 refers to a straightforward motion, following the definitions in Section III.B. The gray level of the gray-scale background indicates the variance of the training data. Note that the plot is a sampled result from a large number of training data. We could see that separating from the center, there are much more green and yellow dots on the right; whereas more red and orange dots on the left. This distribution indicates that the left or right control output are slackly distinguished. The blue dots are all over the space, which indicates that the straightforward motion is universal under different possible image inputs. This behavior was reasonable during the experiments. These clustered points show that the visual feature maps are roughly linear separable in high-dimension space. It intuitively proves that the separable visual features is the basis for generating the corresponding control commands.

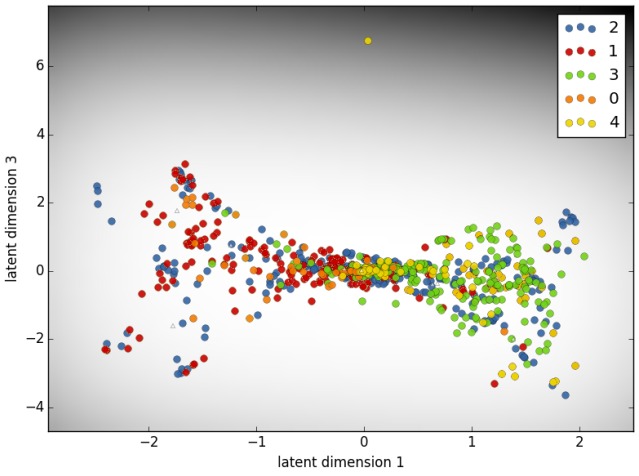


Fig. 5. Visualization of 19200-dimensional feature maps of the training data on a two-dimensional plane.

B. Feasibility to mimic humans for further robotic applications

The exploration behavior is in an active and forward-computing manner, which means that the robot is self-conscious. Unlike geometric models that deals with distance information, our model is trained by taking human instructions as references, and it highly imitates a brain in the similar way to make a decision. In 20 trials, our approach performs perfectly for obstacle avoidance. Traditional obstacle avoidance algorithms requires a local map or a cost

[3]In this work, we empirically chose the projected latent space as $\mathbf{R}^6$.

map to be built before making a decision, which adds to additional computational cost. While in our model, only the direct sensor input is needed and obstacle avoidance task is achieved automatically. This is much like the stress-reaction of a biological neural system. This further indicates that our approach partially simulates a higher-level intelligence.

By further integrated with localization and navigation systems, our model has the potential to become a complete indoor navigation system. A series of tasks, such as visual localization[19], visual homing [20], [21], exploration and path-following[22], could be achieved. We aim to fulfill these tasks by hierarchical networks in the near future.

VI. Conclusion and Future Work

In this paper, we proposed an obstacle avoidance approach in indoor environments based on a complex deep-network structure, and accomplished real-world experiments. Experiments show that our system could successfully manage obstacle avoidance. Comparisons between robot decisions and human decisions showed high similarity. Nevertheless, there are still some limitations, like the offline training strategy is not mostly suitable for robotic applications and a discrete classification may not be precise enough for a continuous state space of the decisions. For the next steps, we will further en-couple on-line learning algorithms with *libcnn* and further extend the target space from discrete space to continuous space.

Materials

Along with this submission, we provide the following additional materials for further bench-marking from the community:

- A novel CNN library named *libcnn* was recently proposed. It emphasizes real-time capability and robotic related applications. It can be obtained at *https://github.com/libcnn/libcnn*
- The data-set with RGB-D input and human operations for exploration is available at *https://ram-lab.com/dataset/rgbd-human-explore.tar.gz* (560 MB). The data-set contains 1104 synchronised RGB-D and joystick information. Further detail is as follows:

Topic Name	msg Type	Description
/camera/depth/image_raw	sensor_msgs/Image	Depth images
/camara/rgb/image_color	sensor_msgs/Image	Colour images
/joy	sensor_msgs/Joy	Joystick commands

References

[1] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[2] D. Maturana and S. Scherer, "3d convolutional neural networks for landing zone detection from lidar." ICRA, 2015.

[3] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," *arXiv preprint arXiv:1412.3128*, 2014.

[4] L. Tai and M. Liu, "A robot exploration strategy based on q-learning network," in *Real-time Computing and Robotics (RCAR) 2016 IEEE International Conference on*, Angkor Wat, Cambodia, June 2016.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[6] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[7] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *In Proceedings of the ACL conference*. Citeseer, 2013.

[8] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631. Citeseer, 2013, p. 1642.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.

[10] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[15] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1915–1929, 2013.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *arXiv preprint arXiv:1411.4038*, 2014.

[17] S. Li, H. Huang, Y. Zhang, and M. Liu, "An efficient multi-scale convolutional neural network for image classification based on pca," in *Real-time Computing and Robotics (RCAR) 2015 IEEE International Conference on*, Changsha, China, June 2015.

[18] Z. Dai, J. Hensman, and N. Lawrence, "Spike and slab gaussian process latent variable models," *arXiv preprint arXiv:1505.02434*, 2015.

[19] M. Liu, D. Scaramuzza, C. Pradalier, R. Siegwart, and Q. Chen, "Scene recognition with omnidirectional vision for topological map using lightweight adaptive descriptors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 116–121.

[20] M. Liu, C. Pradalier, and R. Siegwart, "Visual homing from scale with an uncalibrated omnidirectional camera," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1353–1365, 2013.

[21] M. Liu, C. Pradalier, F. Pomerleau, and R. Siegwart, "Scale-only visual homing from an omnidirectional camera," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3944–3949.

[22] M. Liu, "Robotic online path planning on point cloud," *IEEE transactions on cybernetics*, vol. 46, no. 5, pp. 1217–1228, 2016.