Autonomous exploration of mobile robots through deep neural networks

Tai, Lei; Li, Shaohua; Liu, Ming

Published in:
International Journal of Advanced Robotic Systems

Published: 01/07/2017

Document Version:
Final Published version, also known as Publisher's PDF, Publisher's Final version or Version of Record

Published version (DOI):
10.1177/1729881417703571

Publication details:
Tai, L., Li, S., & Liu, M. (2017). Autonomous exploration of mobile robots through deep neural networks. *International Journal of Advanced Robotic Systems*, *14*(4), 1-9. https://doi.org/10.1177/1729881417703571

Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
July-August 2017: 1–9

DOI: 10.1177/1729881417703571
journals.sagepub.com/home/arx
SAGE

Autonomous exploration of mobile robots through deep neural networks

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

Abstract

The exploration problem of mobile robots aims to allow mobile robots to explore an unknown environment. We describe an indoor exploration algorithm for mobile robots using a hierarchical structure that fuses several convolutional neural network layers with decision-making process. The whole system is trained end to end by taking only visual information (RGB-D information) as input and generates a sequence of main moving direction as output so that the robot achieves autonomous exploration ability. The robot is a TurtleBot with a Kinect mounted on it. The model is trained and tested in a real world environment. And the training data set is provided for download. The outputs of the test data are compared with the human decision. We use Gaussian process latent variable model to visualize the feature map of last convolutional layer, which proves the effectiveness of this deep convolution neural network mode. We also present a novel and lightweight deep-learning library *libcnn* especially for deep-learning processing of robotics tasks.

Keywords

Robot exploration, deep learning, CNN

Date received: 28 July 2016; accepted: 23 February 2017

Topic: Special Issue - Robotic Applications Based on Deep Learning
Topic Editor: Haoyao Chen

Introduction

Background

The ability to explore an unknown environment is a fundamental requirement for a mobile robot. It is a prerequisite for further tasks such as rescue,[1] cleaning[2,3], navigation[4–8] and so on. To achieve this task, a mobile robot should first be able to accomplish obstacle avoidance and then set an exploration plan to efficiently cover the unknown environment. Previous approaches for exploration are mostly based on a probabilistic model and calculated cost maps, like the work of Stachniss et al.[9] The advantage of probability-based models is that they take into consideration uncertainty to describe the real-world cases. However, most of these approaches only utilize geometric information without any cognitive process.[10,11] From the perspective of intelligence, dealing with input directly to generate output without further processing of input information is a kind of low-level intelligence. It will be more satisfactory if a mobile robot could imitate the way human beings deal with such a task. Fortunately, deep learning, with its advantage in hierarchical feature extraction, provides a potential solution for this problem.

Motivation and bio-inspired perception

Deep neural networks, especially those built from artificial neural networks (ANNs), were firstly proposed by Fukushima in 1980.[12] Since the last decade, they have been

[1] Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong
[2] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong

Corresponding author:
Lei Tai, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong 999077, Hong Kong.
Email: onlytailei@hotmail.com

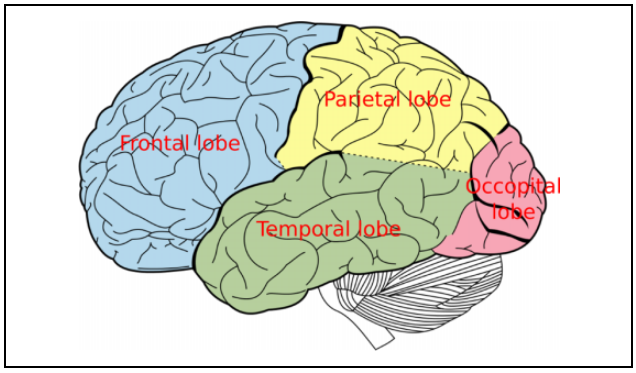

Figure 1. Structure of a human cerebrum, courtesy of Wikipedia.com.

adopted ubiquitously not only in robotics[13–15] but also in natural language processing,[16–19] computer vision[20–22] and so on. When multiple processing layers are used to model a high-level abstraction, the related approaches are generally known as deep learning. Deep learning is a typical bio-inspired technology. It originates from the ANN paradigm,[23] which was proposed in the 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 robotics, 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,[21,22] solved the visual perception as object recognition[24] and solved the acoustic perception. Regarding the decision-making, a recent work by Google DeepMind (http://www.deepmind.com) has shown that the decision-making process could be learned using a deep reinforcement learning model on the Q-functions.[25,26] Note that all these state-of-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 gameplay). 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 article a unified deep network that is able to perform efficient and *human-like* robotic exploration in real time.

From biological point-of-view, both cerebrum and cerebellum are coherently comprised of nervous networks. The perception and action functions are associated with different lobes as shown in Figure 1, such as that the parietal lobe is for spatial sense and navigation. The function of each aforementioned work is equivalent to that of a single lobe of the human brain. This work tries to combine the perception and control with a single deep network. The proposed structure fuses convolutional neural network (CNN)[27] with the decision-making process. The CNN structure is used to detect and comprehend visual features and the fully connected layers are for decision-making. Except for that, no additional modules or algorithms are required for the execution of the task. Overall, we present a complete solution to the autonomous exploration based on a single network structure.

The need of deep learning in robotics

Recently, deep learning techniques have been used for robotics navigation tasks adopted by Sermanet et al.[28] and Hadsell et al.[29] They classify the area in front of the robot for traversability. Giusti et al.[30] also implemented a deep neural network to recognize a forest trail.

Compared with other research fields, robotics research has particular requirements, such as the uncertainty in perception and the demand for real-time operation.[31] 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 of the real-time capability, precision and confidence of judgement is specifically required in robotics. Although there are several libraries for deep learning in computer vision and speech recognition, we still need an ultra-fast and reliable library for robotic applications. As part of the contributions, we hereby present a novel deep-learning library – *libcnn*, 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 article.

Contributions

We address the following contributions of this article:

- We present a novel deep-learning library especially optimized for robotic applications. Its featured modules include scene labelling, object recognition and decision-making.
- We present a deep-network solution towards human-like exploration for a mobile robot. It results in high similarity between the robotic and human decisions, leading 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 network.
- A large indoor corridor data set with human decision label is provided for download.
- The result of a test is compared with human decision quantitatively and visualization of feature maps are shown by Gaussian process latent variable model (GPLVM).

The remainder of the article is organized as follows: the second section 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 the third section. After that, we will introduce the validation experiments of the proposed model in the fourth section. We discuss the pros-and-cons and potential use-cases of the proposed model in the fifth section. At the end, we conclude the article and provide additional reference to related materials. The preliminary experiment of this paper was introduced in.[32]

Related work

The deep network usually functions as a stand-alone component to the system nowadays. For example, Maturana et al. proposed an autonomous unmanned aerial vehicle (UAV) landing system, where deep learning is only used to classify the terrain.[14] Additionally, deep learning has also been used to model scene labels for each pixel, as described in the studies by Farabet et al.[33] and Long et al.,[34] leading to semantic mapping results.

Deep learning for computer vision

Deep learning has been widely used in computer vision for recognition. Considering the receptive field of recognition, these tasks could be categorized into patch-wise classification[35,36] and pixel-wise classification.[37,38] A typical example of patch-wise classification is image classification. Its objective is to assign a label to each image. Canonical data sets for image classification appeared in recent years for validation of new algorithms. These data sets consist of handwritten digits, for example mnist,[39] the street view house numbers, for example svhn data set[40] as well as objects, for example cifar-10.[41] Another example of patch-wise classification is the so-called *object detection, localization and recognition*. In this task, one should first detect the possible locations of an object and then recognize it. One could refer to the ImageNet competition (http://www.image-net.org) for more details for this challenge. As for pixel-wise classification, each pixel is assigned a label of what it describes. A typical application of pixel-wise classification is scene labelling.

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 CNN for feature extraction. In 2013, LeNet5 model was proposed for handwritten digits recognition and highly outperformed the traditional shallow models in terms of recognition accuracy. Since then, numbers of CNN variants have been proposed for feature extraction and representation. Network in network[42] was a model that integrated convolution operation and multilayer perceptron for feature extraction. In our previous work,[43] motivated by the need of real-time computing for robotic tasks, we propose a principal component analysis (PCA)-based CNN model to remove data redundancy in hidden layers, which ensured the fast execution of a deep neural network. Besides these models, a number of regularization algorithms have been proposed. Noticing the coadaptation of neurons in a deep neural network, Srivastava et al.[44] proposed the algorithm of 'Dropout'. In a dropout network, a subnetwork is trained for each iteration during training while an average model is applied when testing. Following dropout, a more generalized algorithm, namely 'Drop-Connect' was proposed. Instead of dropping out nodes in a network, a drop-connect network drops out *connections*, that is, weights of a neural network and proves that the dropping of nodes is just a special case of the proposed network. As for pixel-wise classification problems, Farabet et al.[33] were the first to put CNN into pixel-wise classification tasks. However, their use of patch-by-patch scanning approach was computationally inefficient and a lot of redundant computations were involved. In 2014, a fully CNN was proposed by Long et al.[34] 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 to not only perception but also 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 are 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 in each decision made.

Deep learning for decision-making

Recently, Q-learning models have been adapted to deep neural networks.[25] Mnih et al.[26] 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 perform well when dealing with problems with simple states. While when it comes to problems that require 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.[45] proposed a model-based learning algorithm for planning, which used 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

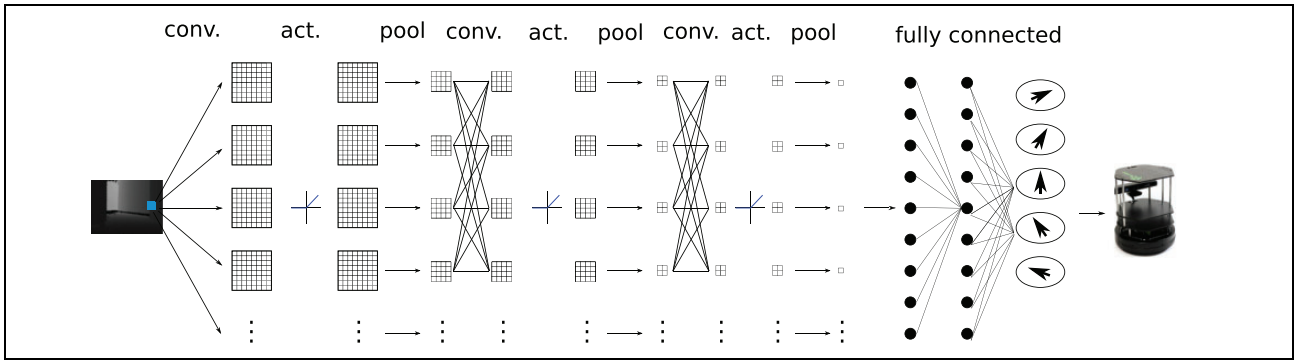

Figure 2. The proposed model that combines CNN with fully connected neural network for robotic exploration. It consists of three convolution-activation-pool layers and two fully connected layers. CNN: convolutional neural network.

is more or less in a simulated environment. When it comes to real-world applications, a lot more factors should be taken into consideration, such as the definition and description of states, the introduction of noise, and so on.

CNN for exploration

In this section, we are going to give a brief introduction to CNN and the proposed model, which is used to generate the control command for exploration of an indoor environment.

CNN preliminaries

CNN is one type of hierarchical neural networks for feature extraction. By back-propagating the gradients of errors, the framework allows learning a multistage feature hierarchy. Each stage of the feature hierarchy is composed of three operations: convolution, nonlinear activation and pooling.

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 ith output feature map. W_i denotes the ith convolution kernel, x is the input and b_i is the ith element of the bias vector, which corresponds to the ith convolution kernel.

Nonlinear activation. After convolution, an element-wise nonlinear 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 $s(x) = 1/(1 + e^{-x})$ and the hyperbolic tangent function $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ were 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.

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.

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

Exploration and confidence-based decision-making

Our CNN model for exploration is illustrated in Figure 2. We use only depth map as the input of our network since depth map provides the most straightforward information of where is traversable. A traditional CNN is adapted for feature extraction, followed by fully connected layers. A weak classifier, the soft-max classifier, is used for classification.

Unlike traditional computer vision applications, where each label of the output represents either an object or scene categories, the outputs of our model are control commands. This is a higher level intelligence compared to the simple task of recognition, since in order to make a decision, there is potential recognition process within the network. The resulted model is a deep network for both recognition and decision-making.

To make a decision and generate control commands, we study the following two approaches by comparison:

1. For the first approach, the output commands are generated by a linear classifier. For this multi-label case, a soft-max classier is used in our model. To achieve this task, the output control commands are sampled and discretized. For the task of object

or scene recognition, where each label represents a specific category, the problem is essentially a discrete classification problem. But for the exploration problem where the robot is supposed to generate control commands, we need to adapt some trade-off. Considering the output state space, the speed and turning are both continuous. Therefore, in order to construct together with a CNN model, this space should be sampled and discretized. However, if too few states are involved in the model, it is highly possible that the robot would oversteer or understeer when it comes to an intersection. While if too many discrete states are going to be classified, it would add up to the difficulty of the classification due to high computational complexity. Here, 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 preset rotational velocities are defined as a discretized angular velocity space, that is, $\vec{\Omega}^* = (\omega_0^*, \omega_1^*, \omega_2^*(=0), \omega_3^*, \omega_4^*)^T$, where ω_i are parameters for discretized control.

For the second approach, we adopt a confidence-based decision-making strategy. We use the same soft-max classifier as previously mentioned. But unlike the first approach, which uses the winner-take-all strategy to make a decision, we use a confidence-based approach. The output of the soft-max classifier could be regarded as the probability of each label, which could also be regarded as the agent's confidence to make such a decision. Notice that it solves the shortcomings of the winner-take-all strategy. For example, the 'winner' possibility is 0.3 and the agent is supposed to take a right turn, while the second highest possibility that tells the agent to go straight forward is 0.29. 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. To solve this dilemma, let c_1, c_2, c_3, c_4, and c_5 denote the confidence of each output label, and ω_a denotes the angular velocity of the mobile robot. The output angular velocity is determined by

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

where $<\ ,\ >$ is an operator of the inner product. Equation (3) is intuitive: if the robot is more confident on certain outputs, it will tend to make the corresponding decisions. Meanwhile, it maintains a trade-off among different output decisions.

Experiments and results

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 ranges from 800 mm to 4000 mm. We use the open source software framework robot operating system (ROS; http://www.ros.org) as the software platform of our system. We simply used a laptop with an Intel Celeron processor without graphics processing unit (GPU) for fast execution of the deep neural network. The system is shown in Figure 3(a). The test environment we used is an indoor environment with corridors insides a building, as shown in Figure 3(b).

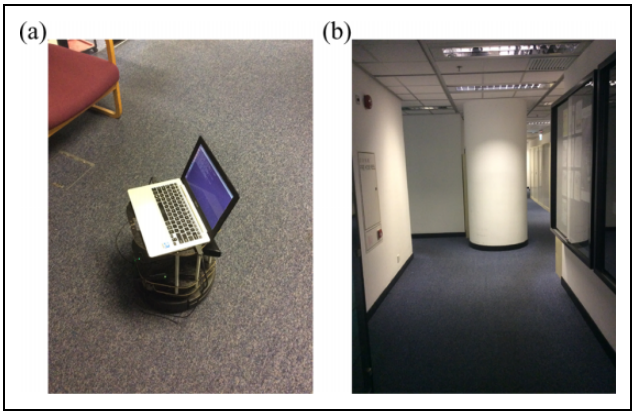

Figure 3. Platform and sample test environment. (a) Hardware and platform. (b) Sample test environment.

Human instruction and 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. By first recording the depth maps received by Kinect and the control commands published by the instructor and then finding the corresponding depth and control commands, we obtain a set of training examples, where the inputs are depth maps and desired outputs are the corresponding control commands. The latter includes speed and steering. Here we point out that the control commands are sampled and discretized to five categories: one for going straight forward, two for turning left with different steering angles, two for turning right and one for staying still, which are corresponding to the defined control labels.

Network configuration

The original depth map size from Kinect is 640×480. In our experiment, the input depth map is first downsampled to 1/4 of the original size, that is, 160×120. This proves to largely reduce the computational cost without introducing many misoperations. The downsampled 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 × 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 is made up of five nodes. The last layer represents the scoring of each output state. The control commands consist of five states: one for going straightforward, two for turning left and two for turning right as previously mentioned. The final decision is calculated by applying the soft-max function to the scores of the five possible states.

Sample results and evaluation

To evaluate the similarity of the agent's decision and human decision, we evaluate our approach in the following two aspects: firstly, the consistency of robot-generated command and reference human-operated command; secondly, the similarity between agents exploration trajectory and human exploration trajectory.

For the consistency test, we show the results using a soft-max classifier for decision-making. We sampled 1104 depth images from the indoor data set where different control categories are almost equally distributed after selection. We use 750 images for training and 354 images for testing. For details of the data set, please refer to the last section.

From Figure 4, we could see that the overall accuracy of the test set is 80.2%. The class accuracy is 79.76%, that is, the mean accuracy of each class. Furthermore, regarding misclassification, there is quite low chance for our system to generate totally opposite decision, for example to misclassify 'left' as 'right'. A large portion of misclassifications could be misclassifying 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.

Figure 5 shows the decisions made by human and robot. In this figure, we plot the angular velocity over time, where positive values mean turning left and negative means turning right. We set linear velocity constant. Here we show two cases, where in the first, there are specific turnings, while in the second case, there are junctions and the environment is much complicated. We sampled 500 points of the curve of human decision and robot decision and 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, together with the figure, show that the robot is able to highly imitate the decision a human makes. 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.

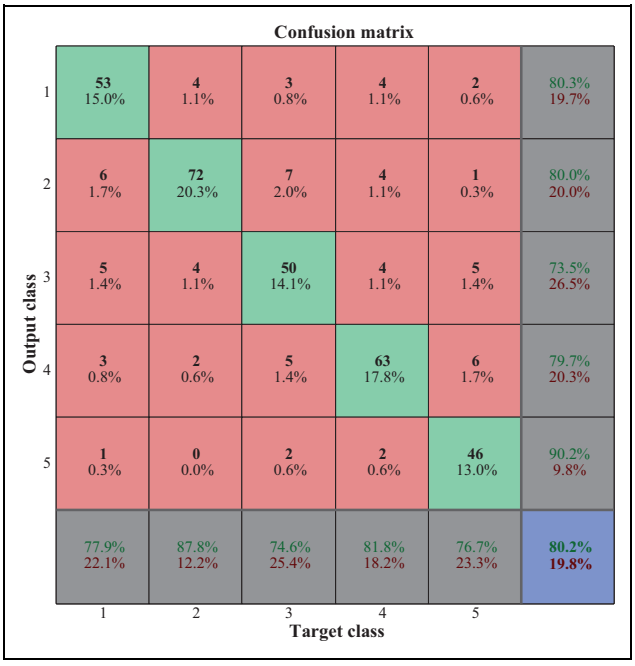

Figure 4. Confusion matrix on the test set. The green-to-red colour-map indicates the accuracy of inference. Note that the outcome is equivalent to a five-labelled classification problem. The result demonstrates outstanding performance of the proposed structure as well as the *libcnn* implementation.

For the test time, the mean time from receiving the input to generate the output command is 247 ms, with variance 12 ms. Note that we get this real-time performance without using GPU for fast execution of the deep network.

Discussion and analysis

In our model, CNN 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 handcrafted features, which allows the agent better adapt to different kinds of environments. Besides, the hierarchical structure of CNN maps the input to a higher dimension and enables the successful application of a linear classifier.

Visualization of feature maps

In order to demonstrate how the trained network functions as an artificial brain, we try to visualize the feature maps generated from the last layer of Figure 2. The second-last layer represents the feature maps, which are further categorized by the last fully connected layer. The outcome leads to the selection of control commands. However, it is not easy to visualize these feature maps

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

where NN is the function of the deep network, I_i is the input depth image with index i and g_i is the corresponding

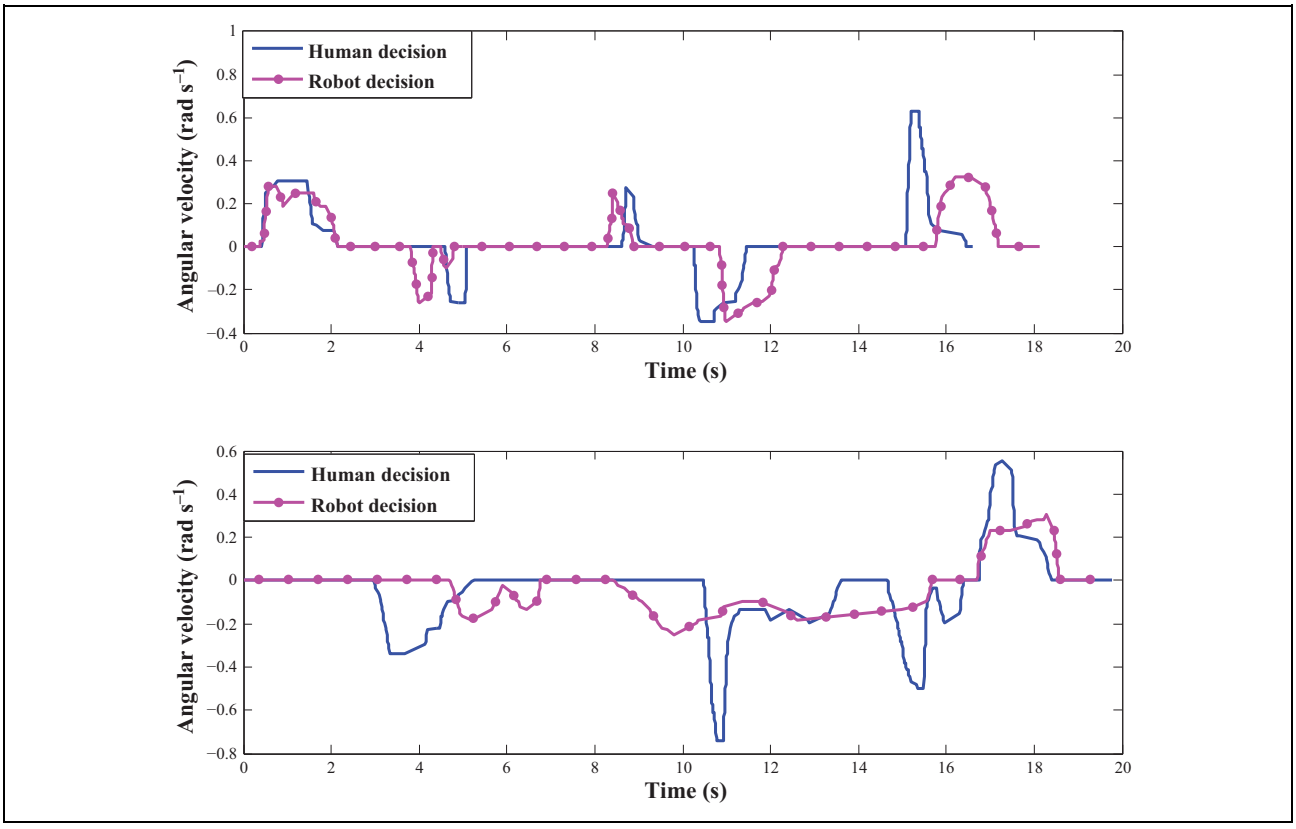

Figure 5. Comparison of human decision and robot decision. 500 points of the curve of human decision and robot decision are collected. The mean absolute difference between the two cases are collected. In the first case, the mean absolute difference is 0.1114 rad/s. In the second case, the value is 0.1408 rad/s.

feature map of the input i. Note that each g_i is with dimension of $g_i \in \mathbf{R}^{19200}$. To solve this problem, we adopt a GPLVM, more specifically, with Spike and Slab Gaussian Process Latent Variable Models (SSGPLVM), which was recently proposed by Dai et al.[46] We project g_i into a lower dimensional space. (In this work, we empirically chose the projected latent space as $\mathbf{R}^6$.) After that, a pair of distinguished dimensions can be visualized as $\mathbf{R}^2$. The final visualization result is shown as Figure 6. The legend indicates the label of the generated control commands, that is, 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 'Exploration and confidence-based decision-making' section. The grey level of the grey-scale background indicates the variance of the training data. Note that the plot is a sampled result, due to a large number of training data. We could see that separating from the centre, there are much greener and yellow dots on the right, whereas more red and orange dots on the left. This distribution indicates that the left or right control outputs are slackly distinguished. The blue dots are all over the space, which indicates that the straightforward motion is universal under heterogeneous image inputs. This behaviour was detectable during the experiments as well.

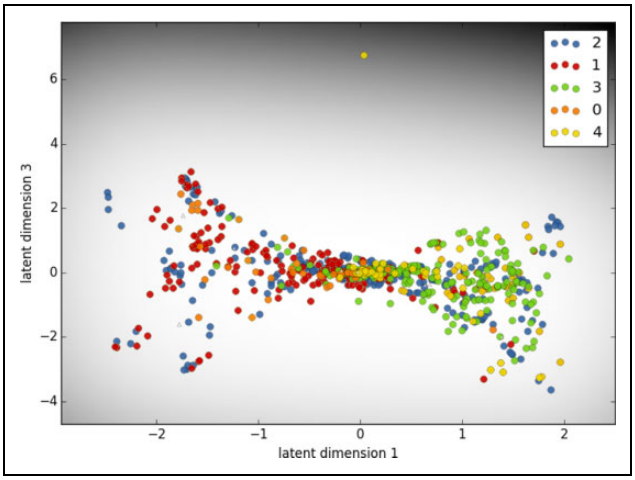

Figure 6. Visualization of 19,200-dimensional feature maps of the training data on a two-dimensional plane.

Feasibility to mimic humans for further robotic applications

As for the experiment, the exploration behaviour is in an active and forward-computing way, which means that the agent is self-conscious. Unlike geometric models that deal with distance information, our model is trained by human instruction,

and it highly imitates human brain in the way to make a decision. Our approach performs perfectly for obstacle avoidance. Traditional obstacle avoidance algorithms require a local map or a cost map to be built before making a decision, which add to the additional computational cost. While in our model, only the visual input is needed and obstacle avoidance task is achieved automatically. This is much like the stress reaction of human beings. This further indicates that our approach 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 serious of tasks, such as visual localization, visual homing,[4] exploration and path-following, could be achieved. We aim to fulfil these tasks all in a human-like way in the near future.

Conclusion and future work

In this article, we proposed a human-like indoor exploration algorithm based on a single deep-network structure and accomplished real-world experiments in typical indoor environments. 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 such as 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 online 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* is 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 http://ram-lab.com/file/rgbd-human-explore.tar.gz (560 MB). The data set contains 1104 synchronized RGB-D and joystick information at http://ram-lab.com/file/lmdb_source_data.tar.gz (4 MB). 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

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. Murphy RR. Human-robot interaction in rescue robotics. *IEEE Trans Syst Man Cybern B Cybern* 2004; 34(2): 138–153.
2. Simoncelli M, Zunino G, Christensen HI, et al. Autonomous pool cleaning: self localization and autonomous navigation for cleaning. *Auton Robot* 2000; 9(3): 261–270.
3. Jager M and Nebel B.Dynamic decentralized area partitioning for cooperating cleaning robots: *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 4, 2002, pp. 3577–3582.
4. Liu M, Pradalier C and Siegwart R. Visual homing from scale with an uncalibrated omnidirectional camera. *IEEE Trans Robot* 2013; 29(6): 1353–1365.
5. Wang H, Guo D, Liang X, et al. Adaptive vision-based leader-follower formation control of mobile robots. *IEEE Trans Industr Elect* 2017; 64(4): 2893–2902.
6. Chen H and Sun D. Moving groups of microparticles into array with a robot -tweezers manipulation system. *IEEE Trans Robot* 2012; 28(5): 1069–1080.
7. Liu M. Robotic online path planning on point cloud. *IEEE Trans Cyber* 2016; 46(5): 1217–1228.
8. Liu M, Colas F and Siegwart R. Regional topological segmentation based on mutual information graphs. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3269–3274. IEEE.
9. Stachniss C, Grisetti G and Burgard W. Information gain-based exploration using rao-blackwellized particle filters. In: *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.
10. Murphy KP. Bayesian map learning in dynamic environments. In: *Conference on neural information processing systems*, 1999, pp. 1015–1021.
11. Murphy K and Russell S. Rao-blackwellised particle filtering for dynamic bayesian networks. In: *Sequential Monte Carlo methods in practice*, 2001, pp. 499–515. Springer.
12. Fukushima K.Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern* 1980; 36(4): 193–202.
13. Lenz I, Lee H and Saxena A. Deep learning for detecting robotic grasps. *Int J Rob Res* 2015; 34(4–5): 705–724.
14. Maturana D and Scherer S. 3d convolutional neural networks for landing zone detection from lidar. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3471–3478.
15. Redmon J and Angelova A. Real-time grasp detection using convolutional neural networks. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1316–1322.

16. Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. *J Mach Learn Res* 2003; 3: 1137–1155.
17. Goldberg Y and Levy O. Word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR, abs/1402.3722*, 2014.
18. Socher R, Bauer J, Manning CD, et al. Parsing with compositional vector grammars. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics 2013*, *Volume 1: Long Papers*, Sofia, Bulgaria, 4–9 August 2013, pp. 455–465. ACL.
19. Socher R, Perelygin A, Wu J, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, October 2013, pp. 1631–1642. Association for Computational Linguistics.
20. Ciresan D, Meier U and Schmidhuber J. Multi-column deep neural networks for image classification. In:*2012 IEEE conference on computer vision and pattern recognition (CVPR)*, 2012, pp. 3642–3649. IEEE.
21. Krizhevsky A, Sutskever I and Hinton GE. Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L and Weinberger KQ (eds) *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, pp. 1097–1105.
22. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.
23. McCulloch WS and Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys* 1943; 5(4): 115–133.
24. Hinton G, Deng Li, Yu D, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Proc Mag* 2012; 29(6): 82–97.
25. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. *CoRR, abs/1312.5602*, 2013.
26. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.
27. LeCun Y, Boser BE, Denker JS, et al. Handwritten digit recognition with a back-propagation network. In: Touretzky DS (ed) *Advances in Neural Information Processing Systems 2, Morgan-Kaufmann*, 1990, pp. 396–404.
28. Sermanet P, Hadsell R, Scoffier M, et al. A multirange architecture for collision-free off-road robot navigation. *J Field Robot* 2009; 26(1): 52–87.
29. Hadsell R, Sermanet P, Ben J, et al. Learning long-range vision for autonomous off-road driving. *J Field Robot* 2009; 26(2): 120–144.
30. Giusti A, Guzzi J, , Cireşan DC, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot Autom Lett* 2016; 1(2): 661–667.
31. Farabet C, Couprie C, Najman L, et al. Learning hierarchical features for scene labeling. *IEEE Trans Pattern Anal Mach Intell* 2013; 35(8): 1915–1929.
32. Jonathan L, Evan S and Trevor D. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.
33. Nair V and Hinton GE. 3D object recognition with deep belief nets. In: *Advances in Neural Information Processing Systems*, 2009, pp. 1339–1347.
34. Hinton GE, Osindero S and Teh YW. A fast learning algorithm for deep belief nets. *Neural Comput* 2006; 18(7): 1527–1554.
35. Shao J, Kang K, Loy CC, et al. Deeply learned attributes for crowded scene understanding. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 4657–4666.
36. Tai L and Liu M. Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not. *CoRR, abs/1612.07139*, 2016.
37. Tai L, Li S and Liu M. A deep-network solution towards model-less obstacle avoidance. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2759–2764.
38. Li H, Zhao R and Wang X. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *CoRR, abs/1412.4526*, 2014.
39. Deng L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Sign Proc Magaz* 2012; 29(6): 141–142.
40. Netzer Y, Wang T, Coates A, et al. Reading digits in natural images with unsupervised feature learning. *NIPS* 2011; 2011(2): 5.
41. Coates A, Ng AY and Lee H. An analysis of single-layer networks in unsupervised feature learning. In: *International conference on artificial intelligence and statistics*, 2011, pp. 215–223.
42. Lin M, Chen Q and Yan S. Network in network. *CoRR, abs/1312.4400*, 2013.
43. Li S, Huang H, Zhang Y, et al. An efficient multi-scale convolutional neural network for image classification based on pca. In: *2015 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, June 2015, pp. 57–62.
44. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 2014; 15(1): 1929–1958.
45. Tani J. Model-based learning for mobile robot navigation from the dynamical systems perspective. *IEEE Trans Syst, Man Cyber B Cyber* 1996; 26(3): 421–436.
46. Dai Z, Hensman J and Lawrence N. Spike and slab Gaussian process latent variable models. *arXiv preprint arXiv:1505.02434*, 2015.