INTECH
open science | open minds

Autonomous Exploration of Mobile Robots through Deep Neural Networks

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 the 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 dataset is provided for download. The outputs of the test date are compared with the human decision. We use Gaussian process latent variable model to visualize the feature map of last convolutional layer and it proves the effectiveness of this deep convolution neural network model. we also present a novel and lightweight deep-learning library *libcnn* especially for deep-learning processing of robotics tasks.

1. Introduction

1.1. Background

The ability to explore an unknown environment is a fundamental requirement for a mobile robot. It is a pre-requisite for further tasks like rescue [1], cleaning [2, 3], navigation [4–8], *etc*. To achieve this task, a mobile robot should first 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 [9]. The advantage of probability-based models is that they take into consideration of 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.

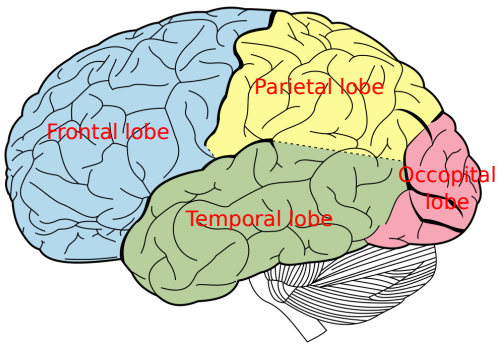

Figure 1. Structure of a human cerebrum, courtesy of Wikipedia.com.

1.2. Motivation and Bio-inspired Perception

Deep neural networks, especially those built from artificial neural networks (ANN) were firstly proposed by K. Fukushima in 1980 [12]. Since the last decade, they have been adopted ubiquitously not only in robotics [13–15], but also in natural language processing [16–19], in 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 origins from the artificial neural network (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] solved the acoustic perception. Regarding the decision-making, a recent work by Google DeepMind (http://www.deepmind.com) has shown that the decision 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 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 *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 Fig. 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 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.

1.3. The need of deep-learning in robotics

Recently, deep learning techniques have been used for robotics navigation tasks adopted by Sermanet and Hadsell [28] [29]. They classify the area in front of the robot for traversability. Giusti [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 paper.

1.4. Contributions

We address the following contributions of this paper:

- We present a novel deep-learning library especially optimized for robotic applications, including scene labelling, object recognition, and decision-making.
- We present a deep-network solution towards human-like exploration for a mobile robot. It results in the 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 dataset 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 GPLVM.

The remainder of the paper is organized as follows: Section 2 goes 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 3. After that, we introduce the validation experiments of the proposed model in Section 4. We discuss the pros-and-cons and potential use-cases of the proposed model in Section 5. At the end, we conclude the paper and provide additional reference to related materials. The preliminary experiment of this paper was introduced in [32].

2. Related work

The deep-network usually functions as a standalone component to the system nowadays. For example, D. Maturana *et. al* proposed an autonomous unmanned aerial vehicle 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 [33, 34], leading to semantic segmentation results.

2.1. 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 the image classification. Its objective is to assign a label to each image. Canonical datasets for image classification appeared in recent years for validation of new algorithms. These datasets consist of hand-written digits, e.g. MNIST [39], the street view house numbers, e.g. SVHN data-set [40] as well as objects, e.g. CIFAR-10 [41]. Another example of patch-wise classification is 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 for more details for this challenge [1]. 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 convolutional neural network (CNN) for feature extraction. LeNet5 [27] 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 multi-layer perception for feature extraction. In our previous work [43], motivated by the need of real-time computing for robotic tasks, we proposed 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 co-adaptation of neurons in a deep neural network, N. Srivastava *et. al* [44] proposed the algorithm of *Dropout*. In a *Dropout* network, a sub-network 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*, i.e. weights of a neural network, and proved that the dropping of nodes is just a special case of the proposed network. As for pixel-wise classification problems, C. 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 computational inefficient and a lot of redundant computations were involved. In 2014, a fully convolutional neural network was proposed by J. Long *et. al* [34] that highly reduces the computation redundancy and could be adapted to large size inputs. In this work, the concept of deep-network is further extended to not only perception but also decision-making tasks.

Although these proposed approaches have been validated on typical datasets, 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 making process.

2.2. Deep learning for decision making

Recently, Q-learning models have been adapted to deep neural networks [25]. V. 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 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.* [45] proposed a model-based learning algorithm for planning, where a 2D laser range finder was used. The model 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 is more or less in a simulated environment. When it comes to real world applications, a lot of factors should be taken into consideration, like the definition and description of states, the introduction of noise, *etc*.

3. Convolutional Neural Network 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.

3.1. CNN Preliminaries

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

[1] http://http://www.image-net.org

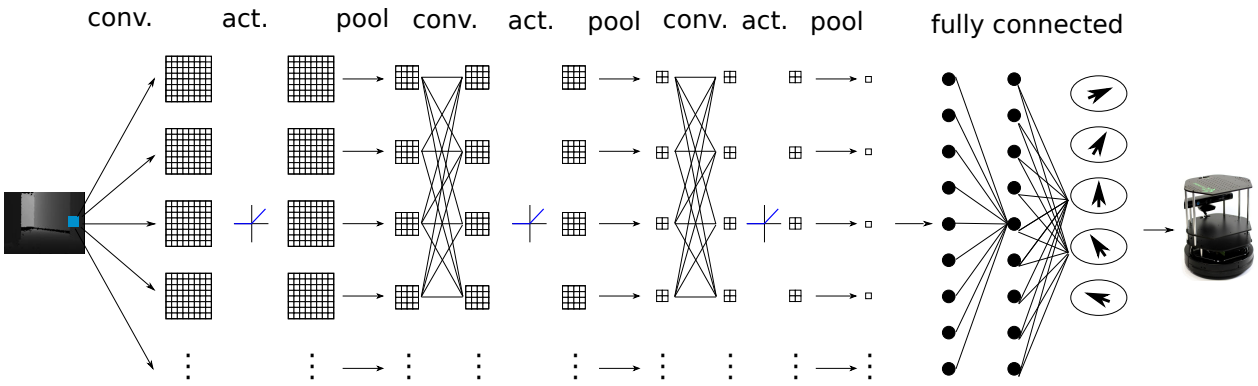

Figure 2. The proposed model which combines CNN with fully-connected neural network for robotic exploration. It consits of 3 convolution-activation-pool layers and 1 fully-connected layer.

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 decreasing of the classification error. The mathematical expression of convolution is denoted as following:

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

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.

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 $s(x) = 1/(1+e^{-x})$ 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 are down-sampled by a factor of s. By introducing the stride parameter, the parameter size of the whole network is reduced.

3.2. Exploration and Confidence-based Decision Making

Our CNN model for exploration is illustrated in Fig. 2. We take only depth image as the input of our network since depth image 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 softmax classifier is used for classification.

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. 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 as comparisons:

1- For the first approach, the output commands are generated by a linear classifier. For this multi-label case, a softmax classifier is used. To achieve this task, the discrete output control commands are sampled. 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 discrete. However, if too few states are involved in the model, it is highly possible that the robot would over-steer or under-steer when it comes to an intersection. While if too many discrete states are going to be classified, it would add up the difficulty of the classification due to high computational complexity. Here we empirically choose five discrete 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 discrete 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 discrete control.

2- For the second approach, we adopt a confidence-based decision-making strategy. We use the same softmax 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 softmax 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, 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 determined by

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

where $< \cdot, \cdot >$ is an operator of inner-product. Equation (3) 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.

4. Experiments and Results

4.1. 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 software framework Robot Operating System(ROS)[2] as the software platform of our system. We simply used a laptop with an Intel Celeron processor without a graphics processing unit (GPU) for fast execution of the deep neural network. The system is shown in Fig. 3(a). The test environment we used is an indoor environment with corridors insides a building, as shown in Fig. 3(b).

(a) Hardware and platform (b) Sample test environment

Figure 3. Platform and sample test environment

4.2. Human Instruction and Data Gathering

In our experiment, we use a set of indoor depth datasets 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 images 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

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

the inputs are depth image 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 classified to 5 categories: one for going straight forward, two for turning left with different steering angles, and two for turning right, which are corresponding to the defined control labels.

4.3. Network Configuration

The original depth image size from Kinect is 640×480. In our experiment, the input depth image is first down-sampled to 1/4 of the original size, i.e. 160×120. This proves to largely reduce the computational cost without introducing many misoperations. The down-sampled depth image is put into a 3-stage "convolution + activation + pooling" cycles, followed by one fully connected layer for feature extraction. The first convolution layer uses 32 convolutional 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 is made up of 5 nodes. The last layer represents the scoring 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 previously mentioned. The final decision is calculated by applying the softmax function to the scores of the 5 possible outputs.

4.4. Sample Results and Evaluation

We sampled 1104 depth images from the indoor dataset where different control categories are almost equally distributed after selection. We use 750 images for training and 354 images for testing. For detail of the dataset, please refer to the last section.

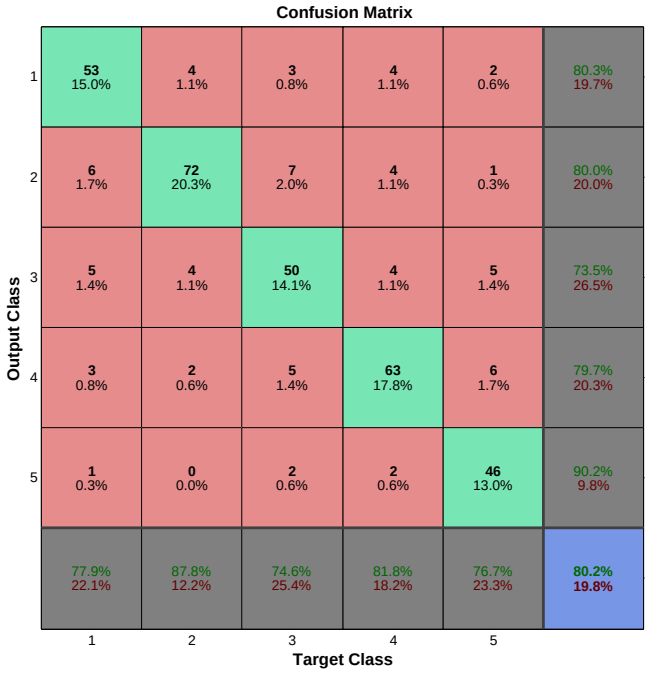

Figure 4. Confusion matrix on the test set. The green-to-red color map indicates the accuracy of inference. Note that the outcome is equivalent to a five-labels classification problem. The result demonstrates outstanding performance of the proposed structure as well as the libcnn implementation.

From Fig. 4, 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 misclassification, there is quite low chance for our system to generate totally opposite decision, e.g. 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.

To evaluate the similarity of the agent's decision and human decision, we evaluate our approach in the following aspect: the consistency of robot-generated command and reference human-operated command.

Fig. 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.

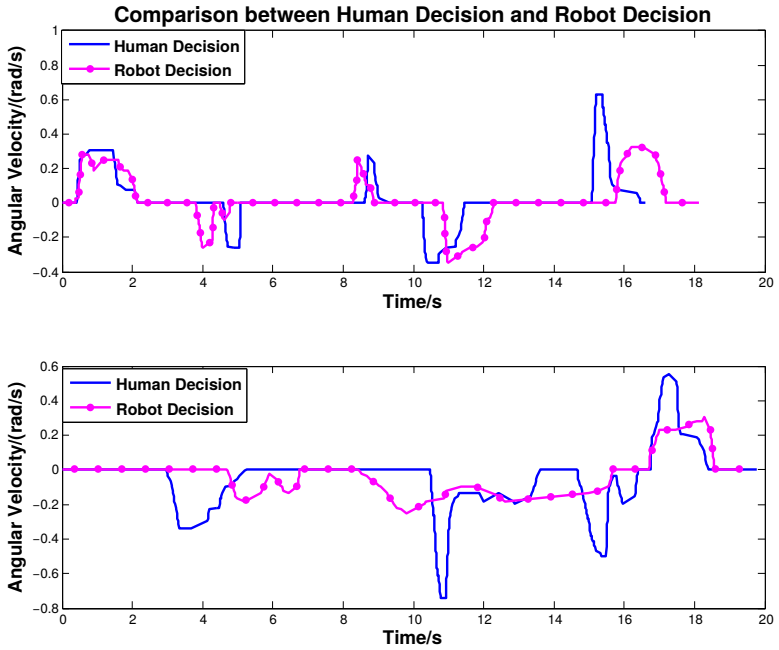

Figure 5. Comparison of human decision and robot decision. 500 points of the curve of human decision and robot decision are collected. The mean absolute difference between the two cases are collected. In the first case, the mean absolute difference is 0.1114 rad/s. In the second case, the value is 0.1408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

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

5. 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 hand-crafted 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.

5.1. 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 Fig. 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.

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

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 this 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. [46]. 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. 6. 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 3. The gray level of the gray-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 center, there are 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 heterogeneous image inputs. This behaviour was detectable during the experiments as well.

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

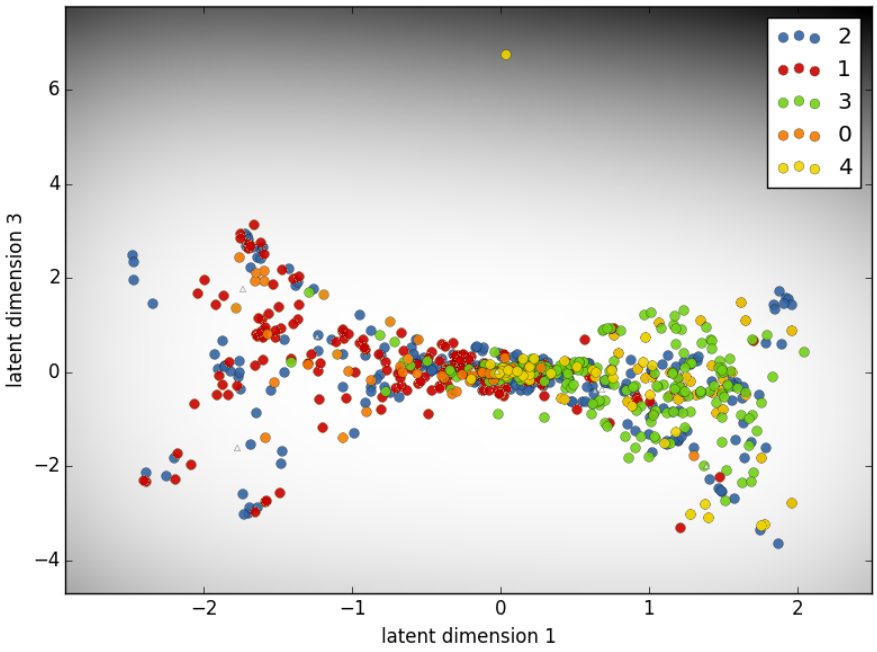

Figure 6. Visualization of 19200-dimensional feature maps of the training data on a two-dimensional plane, through GPLVM. More green and yellow dots are on the right and more red and orange dots are on the left. This distribution indicates that the left or right control output are slackly distinguished.

5.2. 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 adds 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 series of tasks, such as visual localization, visual homing [8], exploration, and path-following, could be achieved. We aim to fulfil these tasks all in a human-like way in the near future.

6. Conclusion and Future Work

In this paper, 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, 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.

7. 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 dataset 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 dataset 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

Acknowledgement

This work was supported by Shenzhen Science, Technology and Innovation Comission (SZSTI) JCYJ20160428154842603 and JCYJ20160401100022706; partially sponsored by the Research Grant Council of Hong Kong SAR Government, China, under project No. 21202816, 16212815, awarded to Prof. Ming Liu.

8. References

[1] Robin Roberson Murphy. Human-robot interaction in rescue robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):138–153, 2004.

[2] Maurizio Simoncelli, Guido Zunino, Henrik I. Christensen, and Klas Lange. Autonomous pool cleaning: Self localization and autonomous navigation for cleaning. *Autonomous Robots*, 9(3):261–270, 2000.

[3] M. Jager and B. Nebel. Dynamic decentralized area partitioning for cooperating cleaning robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, volume 4, pages 3577–3582 vol.4, 2002.

[4] Hesheng Wang, Dejun Guo, Xinwu Liang, Weidong Chen, Guoqiang Hu, and Kam K Leang. Adaptive vision-based leader–follower formation control of mobile robots. *IEEE Transactions on Industrial Electronics*, 64(4):2893–2902, 2017.

[5] Haoyao Chen and Dong Sun. Moving groups of microparticles into array with a robot–tweezers manipulation system. *IEEE Transactions on Robotics*, 28(5):1069–1080, 2012.

[6] M. Liu. Robotic online path planning on point cloud. *IEEE Transactions on Cybernetics*, 46(5):1217–1228, May 2016.

[7] Ming Liu, Francis Colas, and Roland Siegwart. Regional topological segmentation based on mutual information graphs. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3269–3274. IEEE, 2011.

[8] Ming Liu, Cédric Pradalier, and Roland Siegwart. Visual homing from scale with an uncalibrated omnidirectional camera. *IEEE Transactions on Robotics*, 29(6):1353–1365, 2013.

[9] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[10] Kevin P Murphy. Bayesian map learning in dynamic environments. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1015–1021. MIT Press, 2000.

[11] Kevin Murphy and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Sequential Monte Carlo methods in practice*, pages 499–515. Springer, 2001.

[12] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[13] 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.

[14] D. Maturana and S. Scherer. 3d convolutional neural networks for landing zone detection from lidar. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3471–3478, May 2015.

[15] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322, May 2015.

[16] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[17] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[18] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 455–465, 2013.

[19] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

[20] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[22] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[23] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[24] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.
[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
[27] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.
[28] Pierre Sermanet, Raia Hadsell, Marco Scoffier, Matt Grimes, Jan Ben, Ayse Erkan, Chris Crudele, Urs Miller, and Yann LeCun. A multirange architecture for collision-free off-road robot navigation. *Journal of Field Robotics*, 26(1):52–87, 2009.
[29] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.
[30] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.
[31] Lei Tai and Ming Liu. Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not. *CoRR*, abs/1612.07139, 2016.
[32] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2759–2764, Oct 2016.
[33] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.
[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, November 2015.
[35] Vinod Nair and Geoffrey E Hinton. 3d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009.
[36] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
[37] J. Shao, K. Kang, C. C. Loy, and X. Wang. Deeply learned attributes for crowded scene understanding. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4657–4666, June 2015.
[38] Hongsheng Li, Rui Zhao, and Xiaogang Wang. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *CoRR*, abs/1412.4526, 2014.
[39] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
[40] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS workshop on deep learning and unsupervised feature learning*, 2011(2):5, 2011.
[41] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223, 2011.
[42] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
[43] Shaohua Li, Huimin Huang, Yue Zhang, and Ming Liu. 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)*, pages 57–62, June 2015.
[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[45] Jun Tani. Model-based learning for mobile robot navigation from the dynamical systems perspective. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(3):421–436, 1996.
[46] Zhenwen Dai, James Hensman, and Neil D. Lawrence. Spike and slab gaussian process latent variable models. *CoRR*, abs/1505.02434, 2015.