Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
May-June 2018: 1–14

DOI: 10.1177/1729881418777939
journals.sagepub.com/home/arx
SAGE

Toward navigation ability for autonomous mobile robots with learning from demonstration paradigm: A view of hierarchical temporal memory

Xinzheng Zhang[1], Jianfen Zhang[1] and Junpei Zhong[2]

Abstract

Learning from demonstration, as an important component of imitation learning, is a paradigm for robot to learn new tasks. Considering the application of learning from demonstration in the navigation issue, the robot can also acquire the navigation task via the human teacher's demonstration. Based on research of the human brain neocortex, in this article, we present a learning from demonstration navigation paradigm from the perspective of hierarchical temporal memory theory. As a type of end-to-end learning form, the demonstrated relationship between perception data and motion commands will be learned and predicted by using hierarchical temporal memory. This framework first perceives images to obtain the corresponding categories information; then the categories incorporated with depth and motion command data are encoded as a sequence of sparse distributed representation vectors. The sequential vectors are treated as the inputs to train the navigation hierarchical temporal memory. After the training, the navigation hierarchical temporal memory stores the transitions of the perceived images, depth, and motion data so that future motion commands can be predicted. The performance of the proposed navigation strategy is evaluated via the real experiments and the public data sets.

Keywords

Cortical learning algorithm, hierarchical temporal memory, learning from demonstration, navigation, sparse distributed representation

Date received: 14 May 2017; accepted: 27 April 2018

Topic: AI in Robotics; Human Robot/Machine Interaction
Topic Editor: Nak-Young Chong
Associate Editor: Kiju Lee

Introduction

Learning from demonstration (LfD), as an important issue in imitation learning,[1] is a paradigm for robot to learn new tasks. It is inspired from the fact that the human being learns the new skills or obtains the experiences under the guidance of the human experts. In contrast to the traditional scenario, LfD does not require analytically programming a detailed behavior, and allows the users to take the appropriate showing and to "teach" the robot how to perform the new tasks. With observing more demonstrations and

[1] School of Electrical and Information Engineering, Jinan University, Zhuhai, Guangdong, China
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Corresponding author:
Jianfen Zhang, School of Electrical and Information Engineering, Jinan University, 207 Qianshan Road, Zhuhai, Guangdong 519070, China.
Email: eejfzhang@gmail.com

repetitions, LfD provides the robot the ability to acquire the means of behaving new skills. Considering the application of LfD in the navigation issue, a person follows the tour guide to move from any position to the destination when he first visits an unknown place. After the person remembers the path which the guide showed, he learned the navigation skill on how to go to the destination in that place. As the human being learns the navigation behavior above, the robot can also acquire the navigation task via the human teacher's demonstration. This natural communication way between the human teacher and robot learner releases the complex couple of perception and planning in the navigation process, and therefore, LfD for autonomous navigation has become an attractive topic in robotics area.

Related work

The comprehensive surveys on LfD are studied by Argall et al.[2] and Billard et al.[3] These works, respectively, phrased that the LfD can follow the machine learning and computational neuroscience approaches. Nehaniv and Dautenhahn[4] analyzed four key issues of LfD, where "What to imitate" and "How to imitate" in our opinion are the two most important problems for the navigation task.

Learning the relationship between the perceptual information and actions is dominant in the literatures. We call this as end-to-end learning. The difference in previous research works is on the representations of this relationship.

The first paradigm is learning the mapping of the perceptual data and action commands[5–8] directly. To learn this mapping, De Rengervé et al.[5] used artificial neural network to recognize the places according to the panorama. The recognized places combined with odometry and compass data are applied to learn the motion commands by Gaussian mixture models. Similarly, in the study by Choi et al.,[7] leveraged Gaussian process regression, another statistical technique, was presented to get the navigation policy from sequences of sensor data and action pairs. This method also allows demonstrations from casual or novice users not limited to experts. The associations between percepts and actions can be described by a set of fuzzy rules,[6] and predictive sequence learning (PSL) algorithm[6] is used to learn these associations and to predict expected sensor events in response to executed control commands. In addition, with PSL and simulation theory, the robot can generate the experience of novel sequences of events according to the learned relationships.[8]

The second paradigm is to represent the relationship as a planning cost function.[9–12] Learning this cost function is implemented by LEArning to seaRCH algorithm,[9–11] which is a proper technique for imitating a nonlinear cost function, and by Optimal Rapidly-exploring Random Trees planner.[12] Suleman and Awais[13] proposed to find a translatable map function of teachers' and learners' actions by shared circuits model theory. It is a comprehensive and multidiscipline representative theory explaining imitation and other related social functions. Konidaris et al.[14] described a value function as the cost to link the trajectory segments/chains and the sequential motion commands and applied constructing skill tree algorithm which incorporates the pros of hierarchical reinforcement learning and statistical change-point detection algorithm to learn this value function.

To make a summary from the literatures above, the main methods for "How to imitate" are from the statistical theory,[7,12] machine learning,[6,8–11,13] and their incorporations.[5,14]

Why hierarchical temporal memory

As futurist Ray Kurzweil described in his book,[15] the neocortex contains a hierarchy of pattern recognition circuits and they are responsible for most aspects of human thought. He also explains that if there exists a design of the digital neocortex, it could be used to create the same capabilities as the human brain. Hierarchical temporal memory (HTM) theory,[16] first proposed by Hawkins,[17] is an implementation version of Kurzweil's view of digital neocortex. It attempts to model the brain at a functional level rather than at a neuron or molecular level. HTM is a bioinspired model that captures the predominant characteristics of the neocortex. It mimics the neocortex's abilities of learning, inference, and prediction from sequential input patterns that are represented in sparse distributed forms and, therefore, it can describe a complex model of the world. Additionally, HTM uses the sparse distributed representations (SDRs) to represent the complex input data and lend the HTM so much flexibility, which is similar to the idea that the brain is a recursive probabilistic fractal whose line of code is represented within the 30–100 million bytes of compressed code in the genome.[15]

The core of the Kurzweil's book is the pattern recognition theory of mind. Its main idea is that the hierarchical structure is treated as pattern recognizer and is not just for sensing the world but for nearly all aspects of thought. It is natural that HTM was first successfully applied for pattern recognition system.[18–20]

The reasons stated above indicate that HTM can be considered as a promising approach for implementing LfD-based navigation task. Therefore, in this study, we designed an LfD navigation paradigm from the view of HTM. It is also a type of end-to-end learning form. The relationship between perception data and motion commands will be learned and predicted by using HTM. This framework first perceives images to obtain the corresponding categories information; then the categories incorporated with depth and motion command data are encoded as a sequence of SDR vectors. The sequential vectors are treated as the inputs to train the navigation HTM (Nav-HTM). After the training, the Nav-HTM stores the transitions of the perceived images, depth, and motion data so that future motion commands can be predicted. The

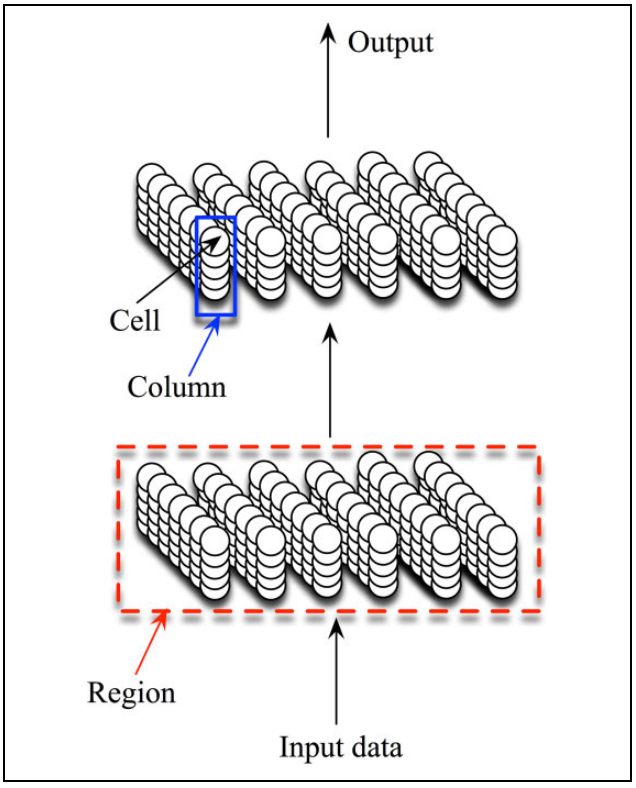

Figure 1. Structure of a typical HTM neural network. HTM: hierarchical temporal memory.

performance of the proposed navigation strategy is evaluated via the real experiments and the public data sets. The contribution of this work is not to appraise the literature above but just to provide a promising solution from the view of mimicking the neocortex capabilities.

Materials and methods

As a memory system, HTM is essentially a type of neural network. It first models the cells, interconnects and arranges cells in columns, organizes columns in a two-dimensional (2-D) array to constitute the HTM region, and finally establishes a hierarchical neural network, as shown in Figure 1. The network learns from the time-varying inputs. These inputs have the format of SDR, which is either transformed from the environmental sensory data by an encoder or received from the outputs of the lower-level region. The HTM network is trained by a simple learning algorithm, namely, the cortical learning algorithm (CLA). It learns and stores sets of distributed input pattern sequences (including the sensory or sensory-motor patterns) and their transitions in the hierarchical organization through spatial and temporal pooling. With the remembered sequences and transitions, the HTM network performs inference (i.e. recognition) and prediction for the new coming inputs. The proposed HTM-based LfD navigation system follows the HTM workflow and is illustrated in Figure 2. The detailed explanation and properties of HTM

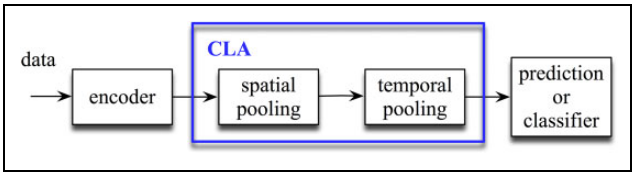

Figure 2. Workflow of an HTM application. HTM: hierarchical temporal memory.

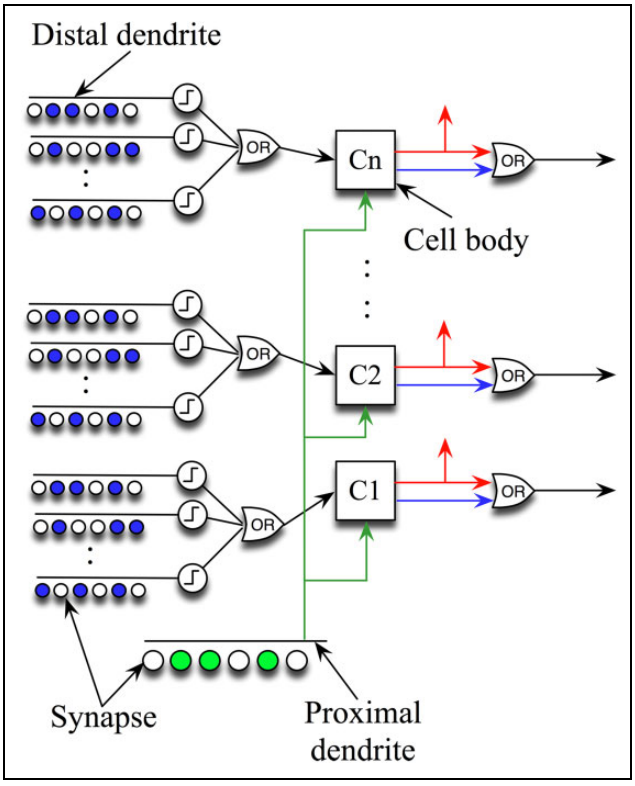

Figure 3. Components of an HTM cell. HTM: hierarchical temporal memory.

and SDR can be found in technique reports.[16] We describe the crucial contents related to our application in the following section.

HTM network model

The HTM network is composed of numerous interconnected HTM cells, which are organized in a column paradigm. HTM cells extract the most important capabilities of biological neurons, and as shown in Figure 3, they have more complex structures than conventional artificial neurons.

A typical HTM cell has three output states: the active state activated from feed-forward input, the predictive state activated from lateral input, and the inactive state. Each HTM cell in one column shares a single proximal dendrite segment (closest to the cell body) and has a list of distal dendrite segments (farther from the cell body). The proximal dendrite segment receives all feed-forward inputs,

including the environmental sensory data and outputs of the lower-level region, via active synapses marked by green dots. These active synapses have a linear additive effect at the cell body. Distal dendrite segments receive the lateral inputs from nearby cells through active synapses marked by blue dots. Figure 3 shows that each distal dendrite segment is a threshold detector. The segment will be activated if the number of active synapses on a segment is above a threshold $\mathrm{Th}_{\mathrm{seg}}$. An OR operation is executed on all active distal dendrite segments to make the associated cell become the predictive state. Synapses of the HTM cells have binary weights and are formed by a set of potential synapses, which are axons that are sufficiently close to a dendrite segment and may become synapses. For the proximal dendrite, a potential synapse consists of a subset of all inputs to a region; and for the distal dendrite, the potential synapses are predominantly from the nearby cells in a region. Each potential synapse is assigned a scalar value ranging from 0 to 1. This scalar value is named as permanence, which represents a closeness or connection degree between an axon and dendrite segment. A larger permanence yields a stronger connection. If the permanence is above a threshold $\mathrm{Th}_{\mathrm{per}}$, the potential synapse becomes a valid synapse, and the weight of this valid synapse is set as 1. The cell body receives the inputs of synapses from proximal and distal segments and provides two outputs along the axon: one is in an active state, which is horizontally sent to other adjacent cells, and the other is the OR of the active and predictive states sent to the cells of the next region.

Because the perception and action are integrated in the HTM network, distal dendritic input can also be the external input. That is, lateral connections between cells will typically be turned off in sensorimotor inference.

Sparse distributed representation

SDR is an efficient information organization in the HTM. Sparse indicates that a small percentage of cells among the large interconnected cells are activated at one time. "Distributed" indicates that active cells are spread out across the region and will be involved in representing the activity of the region.[16] In HTM, the binary SDR converted from a certain encoder is considered because the binary representation is more biologically plausible and highly computationally efficient. Although the number of possible inputs is greater than that of possible representations, the binary SDR does not generate a practical loss of information because the SDR has the following crucial properties.

Semantic overlap: Each cell can be thought of as capturing some "feature" in the inputs; therefore, every active cell in an SDR has semantic meaning assigned from the structure in the inputs. Different active cells at different columns in a region can produce exponential combinations of representation for the various inputs, even if any two inputs look similar. SDR possesses the property of mapping similar inputs to similar representations, which can be identified by comparing the overlap of bits with

$$\mathrm{overlap}\,(x, y) \equiv ||x \wedge y|| \tag{1}$$

where x and y are binary SDRs of input vectors or the stored vectors in a region; $||\bullet||$ is the vector length operator, and it is simply the total number of "1" bits; and $\wedge$ denotes the bit-wise AND operator.

Union: Given a set of SDRs, they can be reliably stored in a single fixed representation by the OR operation following equation (2). This is important for HTM, as it holds a dynamic set of elements and underlies the prediction process in the temporal pooling. As such, a fixed set of cells and connections can operate on a dynamic list, and the union is also used to represent invariance or check a given prediction by searching the union containing its SDR.

$$U_{\mathrm{SDR}} = \bigvee_{i=1}^{k} x_i \tag{2}$$

where $\vee$ is the bit-wise OR operator.

CLA dynamic process

The CLA is a mechanism for explaining the operation in a single region of the neocortex. It has a simple framework and mathematical descriptions. The HTM uses the CLA dynamic process to learn the spatial and temporal variability commonly occurring in sequential input data and then to make predictions. The typical CLA is composed of two subprocesses: spatial and temporal pooling. The detailed explanations are described in the following subsections.

Spatial pooling. The essential function of spatial pooling is to form an SDR of the inputs. When an input appears on a region, each bit in the input signal will be assigned only to a subset of columns. The number of columns is computed by p_{pot}, which is the percentage of inputs that a column can be connected to within a given column's potential radius r_{pot}. The potential synapses associated with cell proximal dendrites on these columns will be activated when their permanence values are above a threshold $\mathrm{Th}_{\mathrm{syn\_per}}$. The number of active synapses is multiplied by a boost factor (bf), which is dynamically determined by how often a column is active relative to its neighbors. This is the phase of overlap, as shown in equation (3)

$$\mathrm{overlap}\,(x_{\mathrm{in}}^{t},\, \mathrm{sdr}_c) = \begin{cases} 0, & \text{if overlap } (x_{\mathrm{in}}^{t},\, \mathrm{sdr}_c) < \mathrm{ol}_{\mathrm{min}} \\ \mathrm{bf}_c \times \mathrm{overlap}\,(x_{\mathrm{in}}^{t},\, \mathrm{sdr}_c), & \text{others} \end{cases} \tag{3}$$

where x^t_{in} is the input SDR vector at time t, sdr_c is the stored SDR in column c, bf_c is the boosting factor for column c, and $\text{ol}_{\min}$ is the minimum overlap.

The columns with the highest activations after boosting disable a fixed percentage of the columns within an inhibition radius. The result of the inhibition is to form a sparse set of active columns that are treated as the inputs of the temporal pooling subprocess in the same region. The mathematical inhibition process is

$$C_{\text{act}}(t) = C_{\text{act}}(t) \cup \{c\}, \quad \text{if overlap}\,(x^t_{\text{in}}, \text{sdr}_c) > 0 \ \text{and overlap}\,(x^t_{\text{in}}, \text{sdr}_c) \geq \text{LA}_{\min} \tag{4}$$

where $C_{\text{act}}(t)$ is the set of the active column index at time t and $\text{LA}_{\min}$ is the minimal number of winning columns.

A Hebbian-like learning procedure is implemented for each of the active columns. Permanence values of synapses aligned with active input bits are increased, and those aligned with inactive input bits are decreased, which is represented in equation (5)

$$\text{pm}^c_{\text{ps}_j} \underset{c \in C_{\text{act}}(t)}{=} \begin{cases} \min(1.0,\ \text{pm}^c_{\text{ps}_j} + \text{pm}_{\text{syn\_inc}}), & \text{if } \text{ps}_j \text{ is active} \\ \max(0.0,\ \text{pm}^c_{\text{ps}_j} - \text{pm}_{\text{syn\_dec}}), & \text{others} \end{cases} \tag{5}$$

ps_j denotes the jth potential synapse in active column c, and its permanence value is denoted by $\text{pm}^c_{\text{ps}_j}$. $\text{pm}_{\text{syn\_inc}}$ and $\text{pm}_{\text{syn\_dec}}$ are the increment and decrement permanence values, respectively. The changes in permanence values make some synapses become valid or invalid accordingly. Simultaneously, the bf and inhibition radius are both updated according to equation (6)

$$\begin{cases} \text{bf}_c = f_{\text{bf}}(\text{ADC}^c_{\text{avg}}, \text{ADC}^c_{\min}) \\ r_{\text{inh}} = \dfrac{(\text{CS}_{\text{avg}} \times \text{PI}_{\text{col}} - 1)}{2} \end{cases} \tag{6}$$

ADC_{avg} (active duty cycle) is a sliding average that represents how often column c has been active after inhibition, for example, over the last 500 iterations. $\text{ADC}_{\min}$ represents the minimum desired firing rate for column c. f_{bf} is the update function, which linearly interpolates the bf between the points $(0, \text{bf}_{\max})$ and $(\text{DC}_{\min}, 1)$, as shown in Figure 4. In general, the bfs for all columns are updated simultaneously. For the inhibition radius updating, the number of inputs to which a column is connected (denoted by CS_{avg}) should first be determined, and then, this number is multiplied by the total number of columns that exists for each input (denoted by PI_{col}). For multiple dimensions, the aforementioned calculations are averaged over all dimensions of inputs and columns.

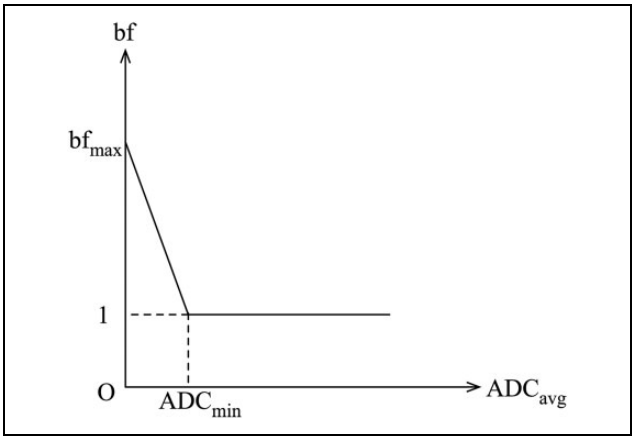

Figure 4. Function for updating the bf. bf: boost factor.

Temporal pooling. The key to CLA is the ability to learn and predict how the patterns in the world change over time and how these changes have a sequential structure that reflects transitions of the real world. The temporal pooling is more complex than spatial pooling because it combines the learning and inference procedures. It consists of three phases, and the inputs are the $C_{\text{act}}(t)$ obtained from the spatial pooling dynamic.

Phase 1: *Determining the active state of cells*. For each active column obtained in spatial pooling, the cells that are fired to a predictive state from a previous time are activated (referring to equation (7)). Simultaneously, the distal dendrite segment on each of these cells is marked as active when the number of synapses is over a threshold Th_{act}. The learning cells are chosen by equation (10). Additionally, if a segment is activated from the learning cells during the previous time, the cell to which this segment connects is set as the learning cell (see equation (8)).

If no cell is in a predictive state, all of the cells in the column are activated, which is defined in equation (9). For this case, the segment that has the largest number of active synapses is found in column c of cell i at time $t-1$, and then, the related cell to which this segment connects is chosen as the learning cell. If no cell has such a segment, we select the cell that has the fewest number of segments as the learning cell (see equation (10)). In phase 1, the resulting set of active cells consists of the current input in the context of prior inputs.

For the perception–action integration case, there is an optional "Learn-On-One-Cell (LOOC)"[21] hysteresis mode. This mode is switched in the following situation. When a column is not predicted but activated by the sensory input, cells that were previously selected as the learning cell would still act as the learning cell at the current time. If no such cell exists, the learning cell is also determined by equation (10). If the LOOC mode is triggered, a copy of the motor signal is added to the input of the distal dendrites

$$\left.\begin{array}{l}\underset{c\in C_{\text{act}}(t)}{\text{na}_i^c}\ (t)=1\\ \underset{c\in C_{\text{act}}(t)}{\text{nl}_i^c}\ (t)=0\end{array}\right\},\ \text{if}\ \underset{c\in C_{\text{act}}(t)}{\text{np}_i^c}\ (t-1)=1\ \text{and}\ \underset{c\in C_{\text{act}}(t)}{\text{sga}_i^c}\ (t-1)=1 \tag{7}$$

$$\left.\begin{array}{l}\underset{c\in C_{\text{act}}(t)}{\text{na}_i^c}\ (t)=1\\ \underset{c\in C_{\text{act}}(t)}{\text{nl}_i^c}\ (t)=1\end{array}\right\},\ \text{if}\ \underset{c\in C_{\text{act}}(t)}{\text{np}_i^c}\ (t-1)=1\ \ \text{and}\ \underset{c\in C_{\text{act}}(t)}{\text{sga}_i^c}\ (t-1)=1\ \text{and}\ \underset{c\in C_{\text{act}}(t)}{\text{sgl}_i^c}\ (t-1)=1 \tag{8}$$

$$\underset{c\in C_{\text{act}}(t)}{\text{na}_i^c}\ (t)=1,\ \ i=1,...,n_c \tag{9}$$

$$\underset{c\in C_{\text{act}}(t)}{\text{nl}_i^c}\ (t)=1,\ \text{if}\left\{\begin{array}{l}\text{cell } i \text{ has the segment with the}\\ \text{largest number of active synapses}\\ \text{OR}\\ \text{cell } i \text{ with the fewset}\\ \text{number of segments}\end{array}\right\}\ \text{at time } t-1 \tag{10}$$

$\text{na}_i^c(t)$ represents the active state of cell i in column c at time t given the current feed-forward input and previous temporal context; $\text{nl}_i^c(t)$ and $\text{np}_i^c(t-1)$ are the learning and predictive state of cell i in column c at time t and $t-1$, respectively; and $\text{sga}_i^c(t-1)$ represents the active segment on cell i in column c at time $t-1$. Similarly, $\text{sgl}_i^c(t-1)$ is the segment activated by the learning cell at time $t-1$. If multiple segments are active, sequence segments are given preference. n_c is the number of cells in column c.

Phase 2: *Forming a prediction based on the input in the context of prior inputs*. Following phase 1, according to equation (11), the cells with active segments are admitted to the predictive state unless they are already active due to feed-forward input. $\text{np}_i^c(t)$ represents the predictive state of cell i in column c at time t. All of the predictive cells form the prediction of the region

$$\underset{c\in C_{\text{act}}(t)}{\text{np}_i^c}\ (t)=1,\ \text{if}\ \underset{c\in C_{\text{act}}(t)}{\text{sga}_i^c}\ (t)=1 \tag{11}$$

On column c of cell i, the current active segment is added to the update list $\text{SU}_i^c(t)$, which will be used in phase 3. To extend the prediction back in time, another distal dendrite segment that has the largest number of active synapses at the previous time is also considered to add to the update list.

Phase 3: *Updating synapses*. Similar to the synapse updates of the proximal dendrite in the spatial pooling dynamic, whenever a distal dendrite segment becomes active, the permanence values of its associated potential synapses are modified by the Hebbian rule only if the cell correctly predicted the feed-forward input. Thus, the synapse permanence values for the segments in update list will be reinforced positively or negatively by

$$\underset{\substack{c\in C_{\text{act}}(t)\\ s_j^i\in \text{SU}_i^c(t)}}{\text{pm}_{s_j^i}^c}\ (t)=\left\{\begin{array}{l}\min(1.0,\ \text{pm}_{s_j^i}^c(t)+\ \text{pm}_{\text{inc}}),\ \ \text{if}\ \text{nl}_i^c(t)=1\\ \max(0.0,\ \text{pm}_{s_j^i}^c(t)-\ \text{pm}_{\text{dec}}),\ \ \text{if}\ \text{np}_i^c(t)=0\ \text{and}\ \text{np}_i^c(t-1)=1\end{array}\right. \tag{12}$$

where $\text{pm}_{s_j^i}^c(t)$ represents the jth synapse permanence value of a segment on column c of cell i, and pm_{inc} and pm_{dec} are the incremented and decremented permanence values in temporal pooling dynamics, respectively.

Finally, a vector representing the OR of the active and predictive states of all cells in a region becomes the input to the next region in the hierarchy. With the prediction, the HTM network can estimate approximately when the inputs

will likely arrive next as well as invoke and separate the motor information.

Results

To examine the performance of the HTM-based navigation strategy, we designed two experiments using the TurtleBot 2 mobile robot in a typical indoor environment of our department. One is a simple navigation in a typical office indoor environment. The robot loaded two motion sensors, odometry and gyro, and moved at translational and rotational speeds of 200 cm s^{-1} and 20° s^{-1}, respectively. The perceptual image data were acquired from a Kinect RGB-D camera loaded on the top of the robot. In these two experiments, we stored RGB images with sizes of 640 × 480 per second. To make the computation efficient, the depth information within a region of interest (ROI) was extracted. The ROIs were selected as individual 64 × 48 rectangles around the image center. Simultaneously, the motion data, including the translational and rotational speeds, were collected from the interior motion sensors. The RGB-D and motion information were incorporated for HTM network training and prediction. The other experiment is designed by using the public data set of outdoor environments to further evaluate our proposed navigation methods.

The HTM was designed based on the open-source project NuPIC (available at https://github.com/numenta/nupic), and its settings were identical for both experiments. The network has a hybrid structure. As shown in Figure 5, the image data were first processed by another vision HTM (VHTM) network, which is an earlier version of HTM implementation, and its output combined with the depth and motion data was encoded to send to the upper one-region Nav-HTM network for motion prediction. We treated the VHTM as a recognition system and set it as a four-region network. Each region has a form of a 2-D cell matrix. The input region has 640 × 480 cells, which is equal to the image size; region 1 is an 80 × 80 cell matrix, region 2 is 10 × 10, region 3 is 2 × 2, and region 4 has only one cell, and it is also the output cell for the recognized category. For the Nav-HTM, the number of cells in each column was set to 32, and the size of the columns was set to 2048 (arranged as 64 × 32 in a 2-D plane). This configuration maintains the diversity of SDR inputs and a low probability of a false match between any two SDR inputs. We applied a scalar encoder[16] to organize the motion data as the two 256-bit one-dimensional (1-D) SDR vector and a custom encoder to represent the depth data as the 8-bit 1-D SDR vector. For the output of VHTM, we also used a scalar encoder to encode the image category as the 16-bit 1-D SDR vector. As shown in equation (13), all encoded 1-D SDR vectors were integrated as a 1024-bit binary string, where the image category and depth bits consisted of perception bits and the wheel velocities were motion bits. This binary string will be sent to the Nav-HTM network for training and prediction.

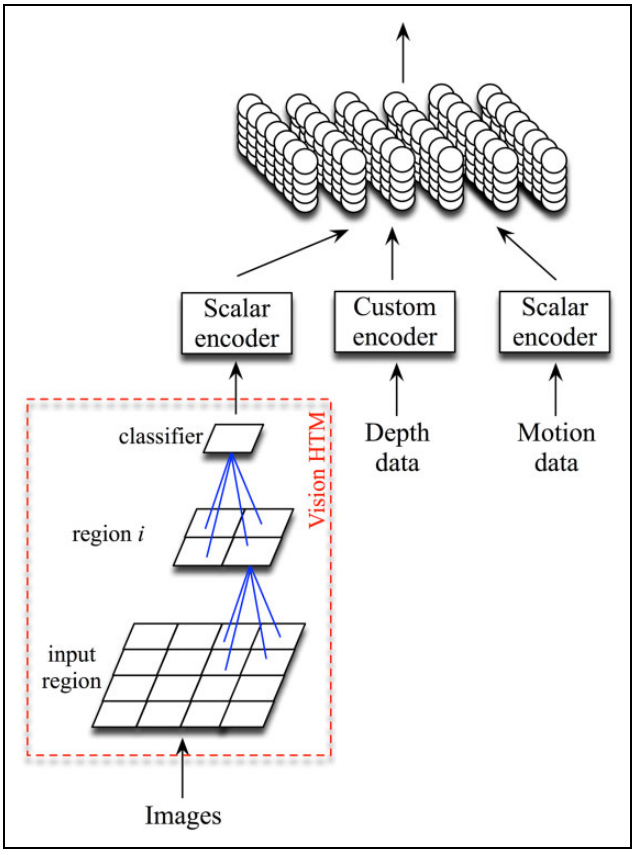

Figure 5. Structure of hybrid HTM network for perception–action application. HTM: hierarchical temporal memory.

$$\underbrace{\overbrace{\times\cdots\times}^{16\text{ bits}}}_{\text{image category}}\ \underbrace{\overbrace{\times\cdots\times}^{8\text{ bits}}}_{\text{depth}}\ \underbrace{\overbrace{\times\cdots\times}^{256\text{ bits}}}_{\text{tran. speed}}\ \underbrace{\overbrace{\times\cdots\times}^{256\text{ bits}}}_{\text{rota. speed}}\ \underbrace{\overbrace{0\cdots0}}_{\text{reserved}} \quad (13)$$

perception bits | action bits

Note: tran. = translational, rota. = rotational.

We set three valid bits of 16 bits for the scalar encoder of the image category. The number of 1s represents the category to which the input image belongs. For example, the encoded SDR *0111000000000000* indicates that the input image is in category 1, whereas *0011100000000000* indicates that it is in category 2. The length of the image category bits is designed for our evaluation cases, and it can be tuned adaptively according to different experimental settings. The custom encoding mechanism for depth bits is determined by the minimal distance extracted within the ROI. If the minimal distance is less than a threshold, that is, 40 cm in our experiments, the least significant bit of the depth bits is set as 1.

The motion bits consist of two groups of speeds on the wheels, one for translational speed and the other for rotational speed. Because the maximal translational and rotational velocities of TurtleBot 2 are 70 cm s^{-1} and 110° s^{-1}, respectively, we set the velocity range for both translation and rotation as [−50, 50] (cm s^{-1} or ° s^{-1}) based on

Table 1. Parameters of the CLA dynamic process.

Parameters	Description	Value
Th_{seg}	Threshold for the number of active synapses on a segment	15
Th_{per}	Threshold for the permanence of potential synapse	0.2
bf_{ini}	Initial value of the bf	1.0
bf_{max}	Maximal bf	2.0
ol_{min}	Minimum overlap	5
r_{inh_ini}	Initial value of the inhibition radius	0
LA_{min}	Minimal number of winning columns	1
r_{pot}	Potential radius, the number of the input bits that are visible to each column	16
p_{pot}	The percentage of the inputs within a column's potential radius to which a column can be connected	0.8
pm_{syn_inc}	Incremented permanence value in spatial pooling	0.05
pm_{syn_dec}	Decremented permanence value in spatial pooling	0.05
Th_{syn_per}	Any synapse whose permanence value is above this threshold will become an active synapse	0.1
ADC_{min}	Minimum active duty cycle	0.001
Th_{act}	Threshold used to determine whether a distal segment is activated	14
pm_{inc}	Incremented permanence value in temporal pooling	0.1
pm_{dec}	Decremented permanence value in temporal pooling	0.1

CLA: cortical learning algorithm; bf: boost factor.

practical considerations, where "−" indicates the negative direction. In our experiments, we defined forward movement and leftward turning as positive for translational and rotational velocities, respectively. Twenty-one bits in each 256 bits of the action encoder are set as valid bits, and 20 cm s^{-1} and 20° s^{-1} are both encoded as the same representation. The reserved bits are designed for the additional sensor information, such as the accelerometer. The CLA dynamic process parameters described in the previous section are listed in Table 1.

The case study on "Department hallway dataset"

In this experiment, a human tele-operated the robot in the corridor by a joystick for demonstration. The robot started to move beside a door and stopped in front of a cabinet in an office room. The robot met three typical objects: an open door, a closed door, and a chair (see Figure 6) during the navigation. We designed a set of simple action strategies: the robot goes through the open door, stops in front of the closed door 40 cm away, and turns left at a distance of 40 cm from the chair. The hand-measured environment map is shown in Figure 7, where the predefined navigation routine is marked by the arrow lines and the robot and several grabbed environment scenes are also displayed. Five sets of data were recorded in two separate demonstrating executions. Each included 140 RGB-D images and motion data. We used the first 140 captured demonstrated data to train the vision and Nav-HTM networks. After the training, the remaining groups of data were sent to the trained networks for offline evaluation. Offline validation is a batch testing, that is, the images collected at all sampling times were first sent to the VHTM to obtain a batch of image category information, then the image categories, depth, and motion data sampled at t_i ($i = 1, \ldots, 139$) were sent to Nav-HTM, and finally the Nav-HTM outputs the predicted inputs of Nav-HTM at t_j ($j = 2, \ldots, 140$). The motion commands can be split from these predictions. The offline evaluation results by using the second demonstrated data set are shown in Table 2 and Figure 8. Table 2 shows that the VHTM outputs for all testing data sets are identical with our desired values, which maintains the valid inputs for the Nav-HTM network. Figure 8 lists one-step ahead sequential action predictions of wheel translational and rotational speeds. It can be found that the predicted commands for the next sampling time are consistent with the practical ones captured by the motion sensors. In particular, when a command switch occurs (highlighted by the black arrows in Figure 8), this prediction mechanism still works well and produces correct motions. These offline examination results demonstrate that our proposed navigation method provides the correct motion predictions according to the different perceived environmental input data.

In online examination, the real-time captured RGB images were sent to the trained VHTM network and the depth data were fed to the Nav-HTM network. Only the motion data taken at the first sampling time were sent to the Nav-HTM network. The Nav-HTM itself predicts a command for the next sampling time according to the current RGB-D and motion data. The predicted action is executed and fed back to the Nav-HTM to integrate with the new RGB-D data so that the next action prediction can be generated. Figure 9 provides the online navigation routine compared with the demonstrated routine. The current routine (marked in red line) recreates the learned routine (marked in blue line). The difference between these two lines is caused by odometer noise and accumulated error of dead reckoning. This result suggests that our proposed approach can be used for online autonomous LfD navigation. In fact, once the robot starts to move, it will maintain velocities received at the initial time, and therefore, the feedback of motion data at every sampling time exactly is used to update the previous actions. The learned motion data in the demonstration process are remembered in the Nav-HTM, and they are treated as the reference for the predicted actions. If the prediction is abnormal, these stored actions can be used for anomaly detection, which will be discussed in the "Conclusion" section.

The computational platform is a Pentium M 1.73 GHz, with a 2G RAM laptop. The time for training the Nav-HTM network is 80.9 s, whereas the VHTM training time is much

Figure 6. Typical objects in the simple experiment setting.

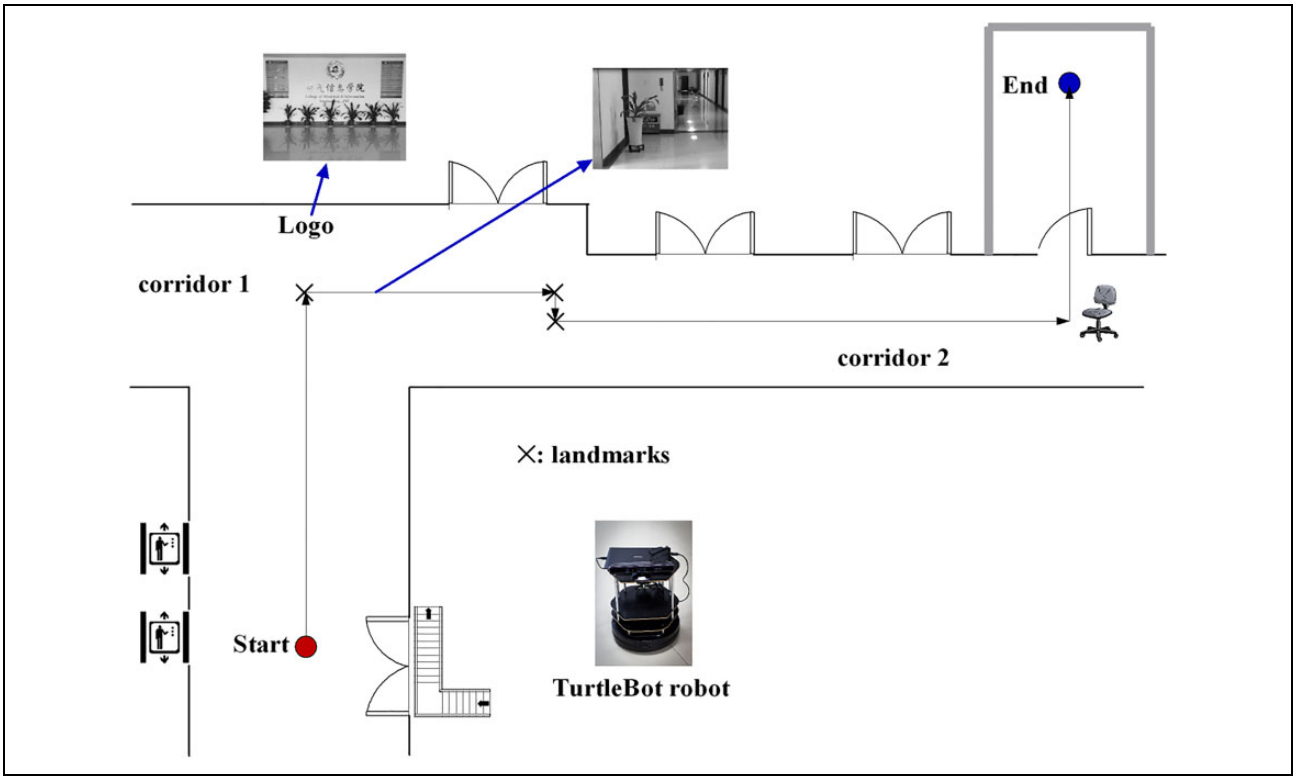

Figure 7. Hand-measured map and predefined navigation routine.

longer (370.7 s). The online evaluation process, which consists of loading trained networks, encoding RGB-D and action data, implementing spatial and temporal pooling, and predicting output, consumes 0.27 s. The cost of validation is considerably less than that of the training because the training is a batch processing. Categorizing all of the RGB images comprises nearly half of the training time. In comparison, only one image frame, depth, and motion data have to be processed in online evaluation; hence, the time cost is reduced considerably. Considering the results in

Table 2. Offline evaluation results for VHTM.

Object	Desired image category (encoded)	Actual image category (encoded)			
		Test data set 1	Test data set 2	Test data set 3	Test data set 4
Open door	0111000000000000	0111000000000000	0111000000000000	0111000000000000	0111000000000000
Closed door	0011100000000000	0011100000000000	0011100000000000	0011100000000000	0011100000000000
Chair	0001110000000000	0001110000000000	0001110000000000	0001110000000000	0001110000000000

VHTM: vision hierarchical temporal memory.

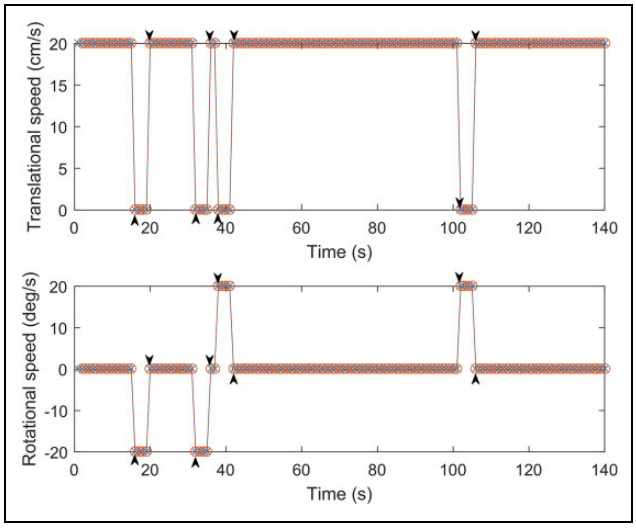

Figure 8. Offline evaluation results of the predicted actions.

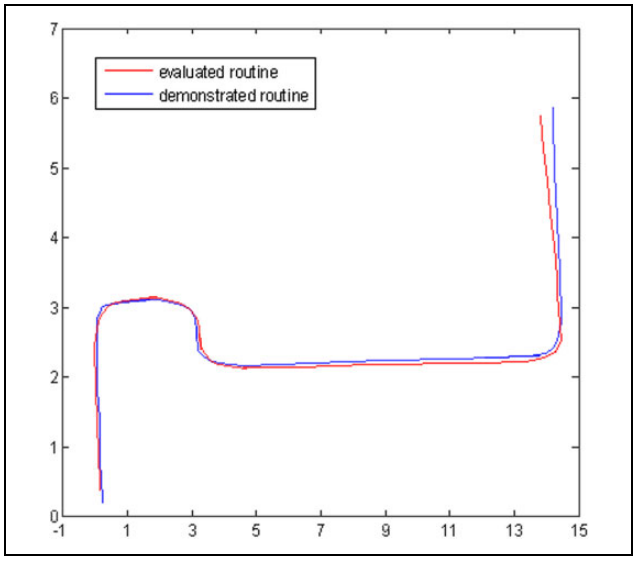

Figure 9. Navigation routine in online evaluation.

terms of computational time, it is logical to use the proposed method for real-time LfD navigation tasks.

The case study on "Barcelona Robot Lab Dataset"

The Barcelona Robot Lab Dataset (this data set is available at http://www.iri.upc.edu/research/webprojects/pau/data sets/BRL/index.php) is applied in this section to further evaluate the performance of the proposed navigation paradigm. This data set is intended to benchmarking algorithms for robust outdoor navigation in robotics community covers 10,000 m^2 of the UPC Nord Campus in Barcelona and include multiple sensor information. The interested data in this article are a time-stamped sequence of action/motion command from the odometry, impressively rich three-dimensional (3-D) laser data, and the sequential stereo images obtained with the custom-built 3-D scanner. Since the trajectories (i.e. the demonstrations) of days 1 and 2 are different, it is not convenient to train the HTM network with the data of day 1 and test the HTM with those of day 2. In this article, we only used the day 1 data to validate our navigation method. The training set is comprised of the data obtained at the odd sampling time ($t_s = 1,3,5,\ldots, n$; $n = 649$, where t_s is the sampling time and n is the total number of data), that is, the training data are selected every two sampling time; in addition, the data corresponding to the motion command switches have to be included in the training set. The stereo images are the inputs of VHTM, the velocities are from the odometry, and the depth is extracted from the stereo images within the ROI 128×96 (the size of original image is 1280×960). After the HTM network is trained, the online motion prediction process, similar to the first experiment, is executed for every sampling time. The difference between this online experiment and the first one is that the image and depth data are not captured in real-time form. We send the stereo images and related depth data to the HTM network frame by frame according to the time stamp. With this configuration, the robustness of our proposed navigation method can be further examined. Figure 10(a) and (b) shows the predicted motion commands compared with the practical commands of data sets. It can be found that there exist errors between the predicted and practical commands which are different from the results in Figure 8. Since, in the first experiment, all the data are used to train and only a part of data are selected as the training set in this experiment, the sequential commands predicted based on the partial demonstration data generate the errors. However, the time interval for training data is short, and especially, the data corresponding to the motion command switches sometimes follow the data grabbed at the odd time sample. This makes the training set almost the continuous data. In online experiment, most motion commands and stereo images have been used in training procedure, and

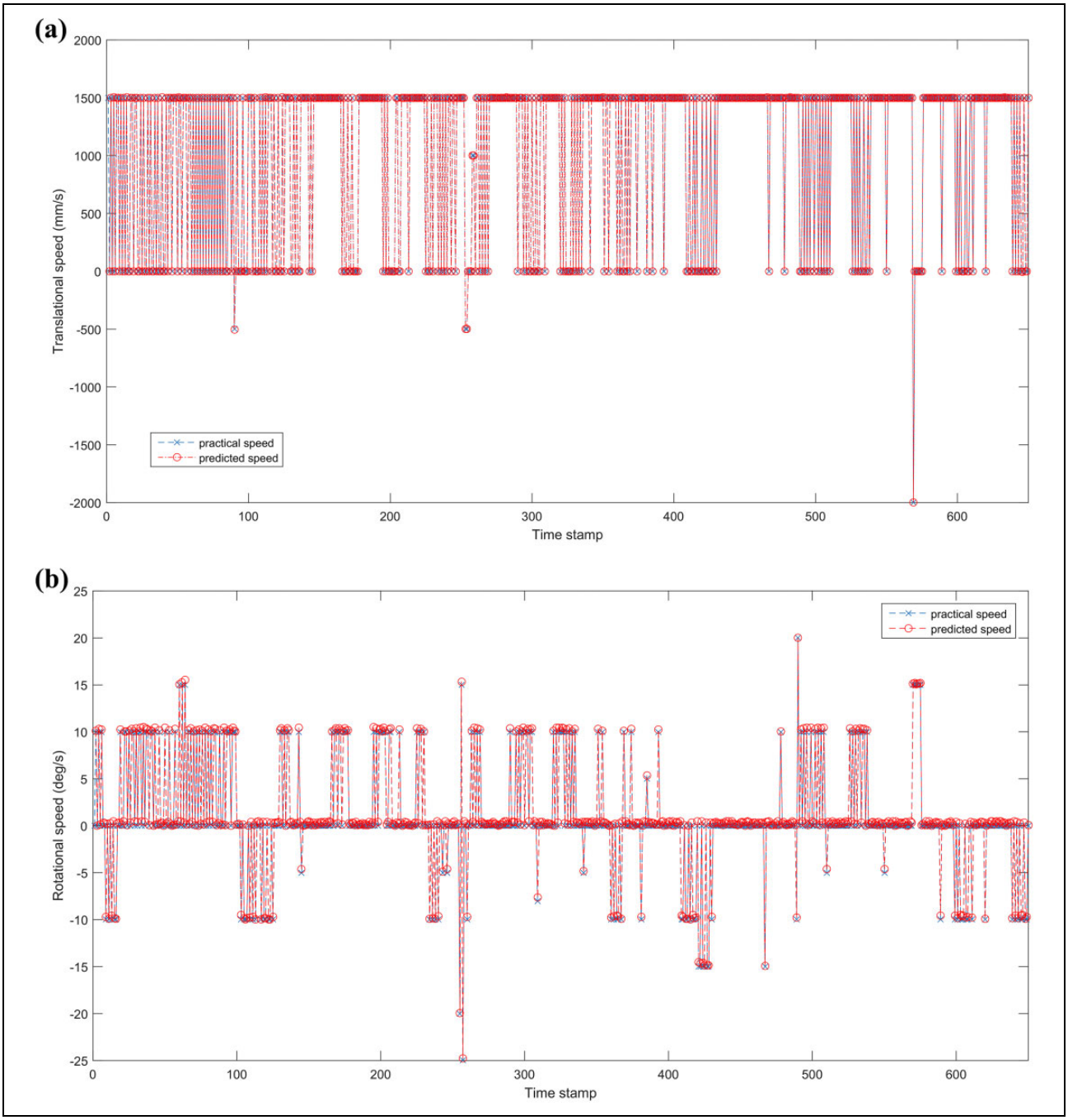

Figure 10. The errors between the practical and predicted motion commands.

the input data sent to HTM networks are recalled from the data set one by one and not the practical data acquired from the real sensors which lack the parameters of sensor uncertainty. Therefore, the calculated robot poses according to the motion commands have small accumulated errors. The mean and variance for the translational and rotational commands are $\mu_{\text{tran}} = 0.0077$, $\sigma_{\text{tran}} = 1.08$ and $\mu_{\text{rot}} = -0.24$, $\sigma_{\text{rot}} = 0.021$, respectively. These errors have little influence on the robot pose estimation, which is illustrated in Figure 11. The predicted navigation routine (dash-dot line with circle marker) is close to the demonstrated robot poses (dash line with cross marker). Table 3 lists the precision and recall rates of our proposed method compared to PIRF-Nav 2.0 algorithm.[22] For the PIRF-Nav 2.0, we used the first motion command to calculate the initial robot pose and then estimated the next pose according to the next motion command and stereo image data. The errors between the estimated pose by using two different methods and practical pose computed from the motion commands of day 1 data set are obtained. With these errors, mean and variance for robot pose can be calculated, as shown in Table 3. The recall rate is the average detection rate at the loop-closure

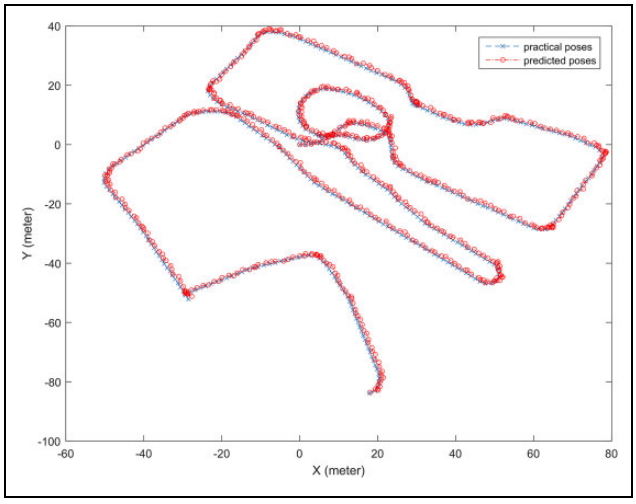

Figure 11. Online evaluation results of the robot poses.

Table 3. Our proposed results compared to PIRF-Nav 2.0.

Method	Robot pose precision		Recall (%)	
	Mean	Variance	LC 1	LC 2
HTM-based navigation	$(-0.481, -0.490, -0.509)^T$	$(0.0817, 0.0816, 0.0852)^T$	91.3	90.7
PIRF-Nav 2.0	$(0.378, 0.383, 0.710)^T$	$(0.174, 0.167, 0.206)^T$	89.1	87.6

HTM: hierarchical temporal memory; LC: loop closure.

parts which is marked in Figure 12. For our proposed method, loop-closure recognition is implemented by VHTM module. From the comparison results, it can be found that the recall rate of our proposed method is a little bit higher than PIRF-Nav 2.0 with the similar robot pose precision. These results state that our proposed LfD navigation can also be applied for an outdoor complex environment.

Discussion

Anomaly detection

There is an important issue to be considered in the online evaluation. If the predicted actions deviate from those expected, the robot likely fails in the autonomous tasks, such as the navigation of our experimental environment. This situation is referred to in the terms of *NuPIC* as an *anomaly*. It is valuable to detect anomalies in real time for many applications. CLA takes the *anomaly likelihood* computed from an anomaly score, a powerful anomaly detection analysis approach, to address this problem.[23] The anomaly likelihood enables the CLA to provide a metric representing the degree to which each record of the input sequence is predictable. It is relative to the data stream rather than an absolute measurement of abnormal behavior and is thus a critical reference to detect whether

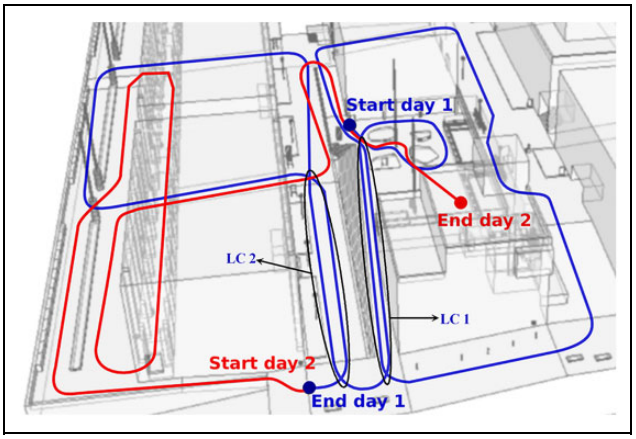

Figure 12. The loop-closure parts of day 1 data set.

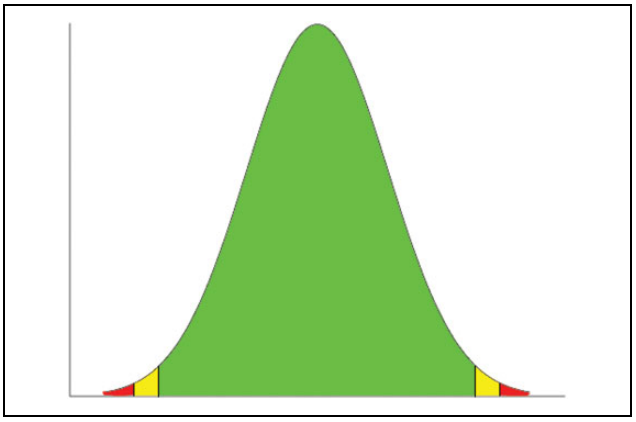

Figure 13. Anomaly likelihood curve.

the pattern with a high anomaly score is actually anomalous. Anomaly likelihood creates an average of the error score and then compares the current average error to a distribution of what the average error has been over the past data stream. This allows us to identify anomalies based on probability. As shown in Figure 13, if the anomaly likelihood is in the green section, this suggests that the record is normal. If it is in the red section, the record shows an abnormal value, which indicates that the pattern is a novel one not seen in any sequence. The yellow section indicates that the pattern is somewhat unusual and that we do not have high confidence. In our application, we consider a pattern anomalous if its likelihood is in the yellow section. Based on the concept of anomaly detection, we calculated the anomaly likelihood for each predicted action in the online navigation experiment. If the anomaly likelihood of any action is above a predefined probability threshold $P_{\text{Th\_ano}}$ (0.90 in our experiment, i.e. the probability or accuracy of the green section is 90%, which is equivalent to a 1.65σ tolerance interval for a normal distribution), we designed a simple action retrieval strategy, that is, recalling the remembered action sequence stored in Nav-HTM to replace that which has

a higher anomaly likelihood. The retrieved action is treated as the prediction for the next time.

We did not detect any abnormal predicted actions in the online navigation experiment above. To validate the performance of the proposed action retrieval strategy, we added an impulse noise with an amplitude of 15 on the 65th predicted translational speed. The anomaly likelihood for this predicted action is 0.954, which is over 0.90. We replaced this anomalous speed with the stored speed and sent it back to Nav-HTM as the prediction for the next time. With this replacement procedure, the following predicted actions after the 65th sampling time were correctly maintained. Because the CLA prediction mechanism in our experiment is one step ahead, we only retrieved one predicted action. If a multistep ahead prediction mechanism is adopted, the number of action retrievals is determined by the number of prediction steps and anomaly likelihoods.

New image encoder

In the present study, we used the earlier generation of HTM implementation to design a VHTM network so that the obtained images could be recognized or classified as a special category, and we further encoded the categories. However, some disadvantages exist for this implementation mechanism. The learning algorithm of the old generation HTM is a partial CLA, which only includes the key CLA components, that is, spatial and temporal pooling, and has simpler learning dynamics. Additionally, the old generation HTM has no concept of encoders, no completed structure of cells, and only one-cell-per-column network. All of these factors negatively impact the learning performance, making this process only suitable for solving the pattern recognition problem. Hence, the VHTM is not an image encoder but rather a classifier system. Additionally, it is a complex programming implementation to incorporate two different generations of HTM under different compiling platforms. In our experiments, we transferred large parts of the old generation HTM code to the new HTM compile platform. However, the compiling platform transformation decreases the computational efficiency.

To address the problems above, it is necessary to design a new encoder to convert the image data to SDR. In our previous work,[24] we attempted to use a visual vocabulary technique to encode the images. Unfortunately, it cannot always maintain the sparse distributed property. A promising work is from Rinkus' research.[25] He proposed a hierarchical sparse distributed coding and quantum computing technique, which has been successfully used to solve the visual processing problem. The future work of our present study can be directed to address how to integrate Rinkus' work into the current CLA algorithm.

Biological evidence for action prediction. The actions incorporated into the perceived inputs are able to contribute to predict the future consequences of the current actions. This is an important cognitive function in the perception–action integration system, which has been examined by Knoblich and Flach.[26] They also proved that this type of prediction becomes more accurate when one obtains the knowledge from one's own actions rather than those of others. Their research provides the biological evidence to support the action prediction mechanism of HTM and its application for robot navigation tasks. However, the current HTM only implements a simple consequence prediction. It provides a sequence of predicted actions, including one-step or multistep predictions, but does not consider the potential information behind these predictions. From a biological viewpoint, the present version of HTM does not link the perceptual input with the action system to predict the future outcome of actions,[26] that is, it does not explain the perception of intentionality for goal-related actions[27] or implement the understanding of the intention hidden in the sequential predicted actions.[28] Additionally, how the predicted actions guide the future perception process is not considered. Therefore, both of these two issues above will be the topics of our future work.

Conclusion

This study is the first attempt to explore the perception–action integration from the view of HTM, which mimics the substantial functions of the human neocortex. The main concept is that sequential perceptual information combined with motion data simultaneously contributes to predicting one-step future actions. The perceived images were first sent to a VHTM network to obtain corresponding categories. The categories were then incorporated with depth and motion data to be encoded as a sequence of 1-D SDR vectors. By using spatial and temporal pooling dynamics of CLA, the sequential vectors were treated as the inputs to train the Nav-HTM network; after the training, the Nav-HTM stored the transitions between the perceived images, depth, and motion so that the future actions could be predicted.

Acknowledgment

The authors thank the NuPIC open-source project and all the contributors of the NuPIC codes.

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) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China under grant number 61203338.

References

1. Schaal S. Is imitation learning the route to humanoid robots? *Trends Cogn Sci* 1999; 3(6): 233–242.
2. Argall BD, Chernova S, Veloso M, et al. A survey of robot learning from demonstration. *Rob Auton Syst* 2009; 57: 469–483.
3. Billard A, Calinon S, Dillmann R, et al. Robot programming by demonstration. In: *Handbook of Robotics*. Berlin, Heidelberg: Springer, 2008, pp. 1371–1394.
4. Nehaniv CL and Dautenhahn K. Like me? Measures of correspondence and imitation. *Cybernetics and Systems* 2001; 32: 11–51.
5. De Rengervé A, D'Halluin F, Andry P, et al. A study of two complementary encoding strategies based on learning by demonstration for autonomous navigation task. In: *Proceedings of the tenth international conference on epigenetic robotics (EpiRob2010)*, Lund university cognitive studies, Lund University, 2010, Vol. 149, pp. 105–112.
6. Billing E, Hellstrom T, and Janlert LE. Simultaneous recognition and reproduction of demonstrated behavior. *Biol Ins Cognit Arch* 2015; 12: 43–53.
7. Choi S, Lee K, and Oh S. Robust learning from demonstration using leveraged Gaussian processes and sparse-constrained optimization. In: *IEEE international conference on robotics and automation, ICRA*, 16 May 2016 – 21 May 2016, Stockholm, Sweden, 2016, pp. 470–475.
8. Billing EA, Svensson H, Lowe R, et al. Finding your way from the bed to the kitchen: reenacting and recombining sensorimotor episodes learned from human demonstration. *Front Robot AI* 2016; 3: 9–25.
9. Bagnell JA, Bradley D, Silver D, et al. Learning for autonomous navigation. *Robot Autom Magaz IEEE* 2010; 17: 74–84.
10. Silver D, Bagnell JA, and Stentz A. Learning from demonstration for autonomous navigation in complex unstructured terrain. *Int J Robot Res* 2010; 29: 1565–1592.
11. Yu CC and Wangi CC. Multi-step learning to search for dynamic environment navigation. *J Inform Sci Eng* 2014; 30: 637–652.
12. Perez-Higueras N, Caballero F, and Merino L. Learning robot navigation behaviors by demonstration using a RRT* planner. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, pp. 1–10. Springer Verlag.
13. Suleman KMU and Awais MM. Learning from demonstration in robots using the shared circuits model. *IEEE Trans Auton Mental Develop* 2014; 6: 244–258.
14. Konidaris G, Kuindersma S, Grupen R, et al. Robot learning from demonstration by constructing skill trees. *Int J Robot Res* 2012; 31: 360–375.
15. Kurzweil R. *How to create a mind: the secret of human thought revealed*. Penguin Books, 2013.
16. Hawkins J, Ahmad S, Purdy S, et al. *Biological and Machine Intelligence (BAMI)*, 16 April 2016. http://numenta.com/biological-and-machine-intelligence/
17. Hawkins J. *On intelligence*. New York: Times Books, 2004.
18. Huang YS and Wang YJ. A hierarchical temporal memory based hand posture recognition method. *IAENG Int J Comput Sci* 2013; 40: 87–93.
19. Du Y, Wang W, and Wang L. Hierarchical recurrent neural network for skeleton based action recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 1110–1118.
20. Rozado D, Rodriguez FB, and Varona P. Extending the bioinspired hierarchical temporal memory paradigm for sign language recognition. *Neurocomputing* 2011; 79: 75–86.
21. Ahmad S. Sensorimotor inference algorithm. https://github.com/numenta/nupic.research/wiki/Sensorimotor-Inference-Algorithm (accessed 2014, 11 Novovember 2014).
22. Kawewong A, Tongprasit N, and Hasegawa O. PIRF-Nav 2.0: Fast and online incremental appearance-based loop-closure detection in an indoor environment. *Robot Auton Syst* 2011; 59: 727–739.
23. Ahmad S and Purdy S. Real-Time Anomaly Detection for Streaming Analytics. *ArXiv e-prints 1607*. http://adsabs.harvard.edu/abs/2016arXiv160702480A (2016, 1 July 2016).
24. Zhang X, Zhang J, Rad AB, et al. A novel mapping strategy based on neocortex model: pre-liminary results by hierarchical temporal memory. In: Presented at the *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 476–481.
25. Rinkus GJ. Sparsey^{TM}: event recognition via deep hierarchical sparse distributed codes. *Front Comput Neurosci* 2014; 8: 160–203.
26. Knoblich G and Flach R. Predicting the effects of actions: interactions of perception and action. *Psychol Sci* 2001; 12: 467–472.
27. Monroe AE, Reeder GD, and James L. Perceptions of intentionality for goal-related action: behavioral description matters. *PLoS One* 2015; 10: e0119841.
28. Blakemore SJ and Decety J. From the perception of action to the understanding of intention. *Nat Rev Neurosci* 2001; 2: 561–567.