Robotica (2012) volume 30, pp. 1013–1027. © Cambridge University Press 2011
doi:10.1017/S026357471100124X

Neuro-fuzzy-based skill learning for robots

Hsien-I. Lin†,* and C. S. George Lee‡

†*Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei, Taiwan*
‡*School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, USA*

(Accepted November 10, 2011. First published online: December 8, 2011)

SUMMARY

Endowing robots with the ability of skill learning enables them to be versatile and skillful in performing various tasks. This paper proposes a neuro-fuzzy-based, self-organizing skill-learning framework, which differs from previous work in its capability of decomposing a skill by self-categorizing it into significant stimulus-response units (SRU, a fundamental unit of our skill representation), and self-organizing learned skills into a new skill. The proposed neuro-fuzzy-based, self-organizing skill-learning framework can be realized by skill decomposition and skill synthesis. Skill decomposition aims at representing a skill and acquiring it by SRUs, and is implemented by stages with a five-layer neuro-fuzzy network with supervised learning, resolution control, and reinforcement learning to enable robots to identify a sufficient number of significant SRUs for accomplishing a given task without extraneous actions. Skill synthesis aims at organizing a new skill by sequentially planning learned skills composed of SRUs, and is realized by stages, which establish common SRUs between two similar skills and self-organize a new skill from these common SRUs and additional new SRUs by reinforcement learning. Computer simulations and experiments with a Pioneer 3-DX mobile robot were conducted to validate the self-organizing capability of the proposed skill-learning framework in identifying significant SRUs from task examples and in common SRUs between similar skills and learning new skills from learned skills.

KEYWORDS: Self-organizing skill learning; Skill decomposition; Skill synthesis; Neuro-fuzzy network; Reinforcement learning.

1. Introduction

Current humanoid robots have been designed to assist and collaborate with humans in performing various tasks. Although they can perform specific tasks using pre-designed programs,[1,2] they are still not as skillful as humans because they lack the ability to self-organize learned skills in learning new skills. In psychology, researchers consider that skill learning is not repeating the actions over and over again but rather a process resulting in a relatively consistent change in behaviors. These changes make behaviors possible to serve as building blocks for skill development, and combining these building blocks creates new skills for new tasks.[3] Thus, skill learning is considered as a process of acquiring skills to achieve a given task and utilizing acquired skills to learn a new skill for accomplishing a new task.

Early robotics research focused on skill acquisition for manufacturing tasks such as assembly,[4] cutting,[5] deburring,[6] etc. To acquire a skill for each different task, a robot needs to be re-programmed to learn the skill for it. Thus, a variety of skill representations and learning algorithms for skill acquisition were proposed to learn skills for different tasks. Among these skill representations, these are dichotomized into non-primitive- and primitive-based representations.

For non-primitive-based representations, these are further divided into local- and global-approximation methods. For local-approximation methods, Albus[7] proposed the Cerebellar Model Arithmetic Computer (CMAC) to acquire robot skills. CMAC is a table-look-up method that reproduces the relation between sensor inputs and system-command outputs. Due to some success of CMAC in acquiring skills, researchers turned their interests to local-approximation methods for acquiring skills. Radial-basis-function network (RBFN)[8,9] is another approach that exhibits locality. Baroglio *et al.*[10] integrated a symbolic interpretation and RBFN to demonstrate that robots exhibited satisfactory performance in a "peg-in-a-hole" task. Although local-approximation methods have shown their success in skill representation, locality impedes the output performance in high-dimensional tasks.

In order to improve local-approximation methods in skill representation, three prevalent global-approximation methods – multi-layer neural networks, fuzzy logics, and Hidden Markov models – were engaged to acquire robot skills. Neural networks are usually trained by a backpropagation algorithm without specific skill models; for example, Nechyba and Xu[11] proposed a neural-network-based method to extract strategies of skills from an expert and provide them to an apprentice; this method was further applied to a one-to-many learning scheme, which is an expert to many apprentices. For fuzzy logic, skill-learning methods are implemented with domain knowledge. Wasik and Safiotti[12] proposed a fuzzy-rule-based control system to learn robot manipulation. They demonstrated that pick-and-place tasks could be realized by a set of behaviors arbitrated by fuzzy rules. Yang *et al.*[13] and Hovland *et al.*[14] considered that human actions might possess inherent stochastic property and employed hidden Markov models to acquire human skills.

For primitive-based skill representations, many robot skill-learning systems adopted motion primitives,[15,16] perceptual-motor primitives,[17] motor schemas,[18] motor programs,[19]

* Corresponding author. E-mail: sofin@ntut.edu.tw

or behavior-based systems[18,20,21] to perform various tasks. These primitives serve as basic units to perform specific tasks. For example, Speeter[15] defined a set of motion primitives of a Utah/MIT Dextrous Hand, such as open, pinch, rotate, swing, etc., to perform manipulation tasks. Matarić *et al.*[22] implemented skill imitation through learning behaviors such as reaching, bouncing, waving, or swinging by discrete straight lines and continuous oscillatory movements. For behavior-based systems, skills can be acquired by constructing a network of behaviors. Behaviors are used to take account of the relationship between stimuli and responses and encapsulate detailed actions to represent motion patterns such as avoiding obstacles, wandering, reaching, etc. They are usually encoded by reactive rules[20,23,24] or mathematical formalism.[18,25,26]

Other than skill acquisition, it is also important to clearly address how new skills are conceived from learned skills. Not surprisingly, researchers employed machine-learning techniques to learn new skills from learned skills. They assumed a new skill could be learned by "black-box" primitives because there is no skill representation described for "black-box" primitives, and there is also no need to worry about how the new skill is represented by the skill representation of "black-box" primitives. One of the prevalent methods for learning new skills by using "black-box" primitives is to plan them in a sequence. However, in a real-world situation, incomplete knowledge of tasks impedes the determination of the sequential order of "black-box" primitives. To solve this problem, reinforcement learning is often adopted.[27,28] Later, to learn skills for complex tasks by "black-box" primitives, hierarchical reinforcement learning approaches were introduced. Among these learning approaches, Sutton's option formalism,[29] Parr and Russell's hierarchies of abstract machines (HAMs) approach,[30] and Dietterich's MAXQ framework[31] are the most notable ones.

Although previous skill learning approaches have demonstrated some success in skill acquisition, it is not easy for them to reuse their skill representations and learning algorithms to learn new skills. For non-primitive-based methods, they do not have good modularity and reusability of learned skills for learning new skills because their skill representations are designed to learn particular skills. On the other hand, primitive-based methods have good modularity and reusability in learning new skills, but they are designed manually and take much domain knowledge to design primitives that may not be autonomously adjustable to various situations of a task. In addition, even though the "black-box" representation is used to demonstrate its modularity and reusability across problems, it lacks the description of detailed stimuli–actions relationship encapsulated in the primitives. Thus, it is still difficult to use the "black-box" representation to realize skill learning in a robot system.

By integrating the advantages of non-primitive- and primitive-based methods of skill representation, this paper proposes a neuro-fuzzy-based, self-organizing skill-learning framework to obtain modularity and autonomy of skill representation in a coherent manner. The proposed framework is distinguished by its capability of self-categorizing skills into significant stimulus-response units (SRUs) and self-organizing learned skills composed of SRUs into new skills; it provides robots with the ability of skill learning from task examples instead of manually designed skills. The term, self-organizing, loosely means that a robot can develop a new skill from learned skills without human intervention. The proposed neuro-fuzzy-based, self-organizing skill-learning framework can be realized into two phases – skill decomposition and skill synthesis. Skill decomposition aims at representing and acquiring skill by SRUs. Skill synthesis aims at self-organizing a new skill by sequentially planning learned skills composed of SRUs. In this regard, learned skills represented by SRUs can be self-organized for learning new skills.

A stimulus-response unit, defined as a mapping from the domain of perceptual stimuli to the domain of action responses, is the building block used in our proposed skill-decomposition and skill-synthesis framework. For particular experiments, stimuli are the sensor input variables and responses are the action output variables. They are appropriately chosen to characterize a skill. The domain of stimuli is defined as the set of independent sensor input variables of a skill. The domain of responses is defined as the set of independent action output variables of a skill.

This paper is organized as follows. In Section 2, the proposed neuro-fuzzy-based, self-organizing skill-learning framework is described. In Section 3, skill decomposition is introduced, and a neuro-fuzzy network with supervised learning is proposed to represent a skill and categorize it into SRUs. Then resolution control is introduced and utilized to prune unnecessary SRUs. The reinforcement learning is proposed to verify the sufficiency of the SRUs obtained from resolution control for accomplishing a given task. In Section 4, skill synthesis is introduced, and common SRUs from similar skills are determined and verified. In addition, self-organizing a new skill from learned skills with reinforcement learning is introduced. In Section 5, computer simulations and experiments on a Pioneer 3-DX mobile robot to validate the performance of the proposed neuro-fuzzy-based, self-organizing skill-learning framework are discussed, and conclusions are summarized in Section 6.

2. Proposed Neuro-Fuzzy-Based, Self-Organizing Skill-Learning Framework

The proposed self-organizing skill-learning framework is shown in Fig. 1, which can be conveniently realized in two phases – skill decomposition and skill synthesis. From a robot-control architecture point of view, skill decomposition is to acquire a skill and decompose it into SRUs that deal with the mapping between stimuli and responses in the low-level control; skill synthesis is to sequentially plan learned skills that are composed of SRUs for accomplishing a task in the high-level planning.

Skill decomposition. In the first phase, skill decomposition acquires a skill from a human and decomposes it into a number of significant SRUs, and this process is realized in three stages. In the second phase, skill synthesis learns new skills by synthesizing acquired skills that are composed of SRUs, and this process is realized in two stages. In the

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

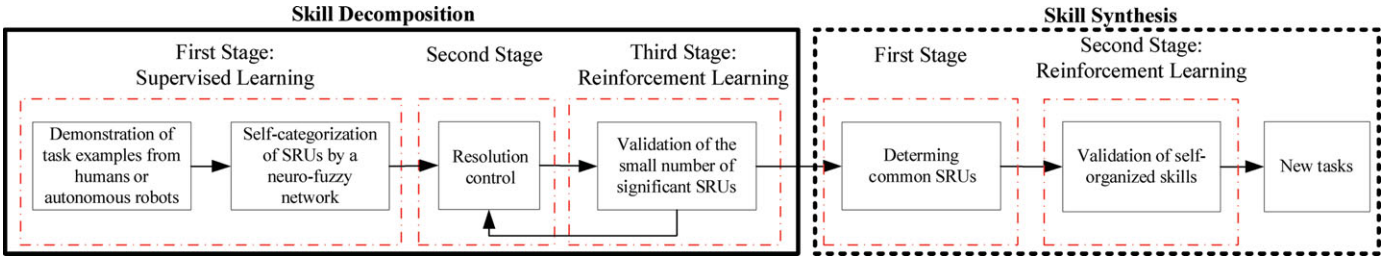

Fig. 1. (Colour online) The proposed neuro-fuzzy-based, self-organizing skill-learning framework. Solid line: skill decomposition; dashed line: skill synthesis.

first stage of skill decomposition, a five-layer, neuro-fuzzy network acquires a skill and categorizes it into a number of initial SRUs that capture the behavioral patterns of the skill as demonstrated by a human. Mathematically, each SRU is regarded as a fuzzy rule in the neuro-fuzzy network. By using human demonstration data, supervised learning is performed to tune the parameters of the neuro-fuzzy network for generating the same skill as done by a human. Since some of these initial SRUs may be unnecessary due to extraneous actions demonstrated by a human, resolution control is proposed to prune unnecessary SRUs and generate a sufficient number of significant SRUs in the second stage. In the third stage, reinforcement learning with an adaptive-heuristic-critic algorithm is utilized to validate the sufficiency of significant SRUs for accomplishing the given task. If not, their behaviors are autonomously adjusted in the reinforcement-learning stage until the task is accomplished within an acceptable number of trials.

In skill decomposition, there are merits of representing a skill by a neuro-fuzzy network. First, a neuro-fuzzy network-based skill representation provides low-level, connectionist-learning capability and high-level, fuzzy IF-THEN-rule thinking. Thus, skills as represented by SRUs can be easily learned and adjusted by using a neuro-fuzzy-based supervised learning from task examples. Second, the connectionist structure of a neuro-fuzzy network-based skill representation provides a mechanism for different types of learning (e.g., supervised and reinforcement learning) because connections in a neuro-fuzzy network can propagate and memorize signals. Thus, the benefits of supervised and reinforcement learning can be appropriately combined and realized in a neuro-fuzzy network-based skill representation. Third, after a skill has been learned, its fuzzy rules can be extracted and expressed explicitly from the SRUs of a neuro-fuzzy network.

Resolution control in skill decomposition aims to determine the number of significant SRUs from the initial SRUs categorized from a neuro-fuzzy network. These significant SRUs enhance generality rather than fidelity of the initial SRUs. Thus, some extraneous actions caused by unnecessary SRUs can be removed. Resolution control is achieved by decreasing the resolution of perceptual stimuli of SRUs and averaging out their corresponding action responses. In other words, resolution control ensures the situations excited by similar perceptual stimuli have their average action responses. By doing this, the number of SRUs decreases. Even though the reduced number of SRUs sacrifice the fidelity of the initial SRUs, they obtain the generality of the initial SRUs by simplifying the underlying mapping from the domain of perceptual stimuli to the domain of action responses for accomplishing a given task.

Once resolution control generates candidates of the significant SRUs, the reinforcement learning in skill decomposition validates them for accomplishing the given task. By using a binary critic, the reinforcement learning adjusts the actions of SRUs by tuning their membership functions of perceptual stimuli and action responses. The learning process continues until the given task is accomplished. In skill decomposition, the curse of dimensionality of the reinforcement learning is eased by using the SRUs as learned by the supervised learning in a neuro-fuzzy network.

Skill Synthesis. In the second phase, skill synthesis aims to learn new skills for new tasks by synthesizing the acquired skills composed of SRUs and is realized in two stages. In the first stage, common SRUs between two similar skills are determined. The common SRUs can also be utilized to keep the pool of SRUs to a minimum. In the second stage, it focuses on self-organizing new skills for new tasks by utilizing learned skills composed of SRUs and creating new SRUs if necessary, and validating the sufficiency of the self-organized new skills by reinforcement learning. The reinforcement learning in skill synthesis adopts a Q-learning algorithm instead of the adaptive-heuristic-critic algorithm used in skill decomposition.

The purpose of determining common SRUs in skill synthesis is to measure the similarity between two skills and keep the pool of SRUs to a minimum. Since a skill is composed of significant SRUs, and each significant SRU is represented by a fuzzy rule, it is possible to determine the similarity between two skills by developing a similarity measure to measure the similarity of the fuzzy rules of SRUs. The success of determining common SRUs provides a quantitative similarity measure between skills and keeps the pool of significant common SRUs to a minimum.

Skill synthesis attempts to learn new skills by self-organizing learned skills, and creating new SRUs if a robot has difficulties in some situations to accomplish a given task by using learned skills. Since a task can be decomposed and represented by a sequence of subtasks, and each subtask can be accomplished by a learned skill, then learning a new skill can be done by planning the execution sequence of learned skills for a new task. To learn the sequential planning in skill synthesis, we adopt a Q-learning algorithm because it is more effective than the adaptive-heuristic-critic algorithm used

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

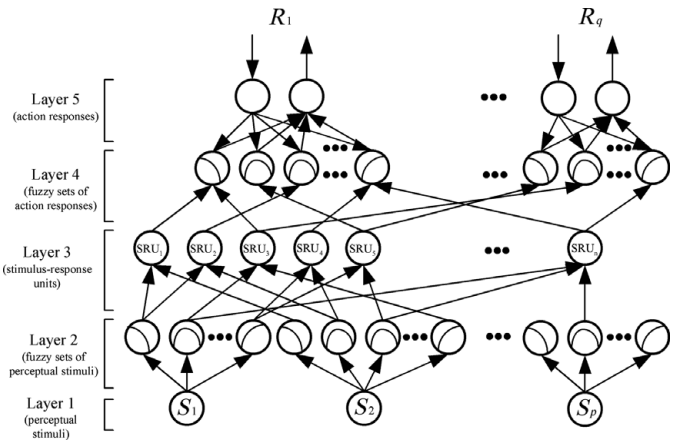

Fig. 2. Skill representation is realized by a five-layer neuro-fuzzy network (S: perceptual stimulus; R: action response; SRU: stimulus-response unit).

in skill decomposition. Other than the sequential planning of learned skills, action responses of learned skills are performed concurrently because a skill is composed of SRUs that are concurrently executed. Thus, in skill synthesis, new skills are learned by a means of both sequential and concurrent executions of action responses of SRUs.

3. Skill Decomposition Phase

3.1. Neuro-fuzzy-based skill representation with supervised learning

Researchers in psychology[3,32,33] and engineering[34–36] have defined skill from different perspectives. The consensus from the previous literature is that a skill is an ability to accomplish a task. To make a skill implementable, we define that a skill is a stimulus–response mapping from the domain of stimuli to the domain of responses and can be represented by a set of SRUs. Here, the proposed skill representation is for learning motor skills instead of arbitrary ones. Thus, we propose to mathematically represent a skill as a fuzzy-rule-based system that is expressed as

$Rule = \{Rule^1, Rule^2, \ldots, Rule^i, \ldots, Rule^n\}$,
$Rule^i$: IF(S_1 is T_{s1} and $\cdots$ and S_k is T_{sk} and $\cdots$ and S_p is T_{sp}), THEN (R_1 is T_{r1} and $\cdots$ and R_k is T_{rk} and $\cdots$ and R_q is T_{rq}),

where $Rule^i$ is the i^{th} fuzzy rule, T_{sk} is a fuzzy set of S_k (a perceptual stimulus), T_{rk} is a fuzzy set of R_k (an action response), and p and q are the numbers of perceptual stimuli and action responses, respectively.

This fuzzy-rule-based system can be realized by a five-layer, neuro-fuzzy network (see Fig. 2).[37,38] Each SRU is mathematically regarded as a fuzzy rule in the neuro-fuzzy network. We shall describe the functions of the nodes in each of five layers to note their representation of a skill. The details of the net input and activation function of nodes in each layer are described in detail in Lin and Lee's book.[38]

Layer 1: Nodes in this layer represent perceptual stimuli.

Layer 2: Each node in this layer represents a fuzzy set belonging to a perceptual stimulus. Each stimulus has links with the nodes representing its fuzzy sets. The membership functions of the fuzzy sets are bell-shaped functions with two parameters: mean and variance. The number of nodes belonging to a perceptual stimulus is decided by either prior experience or by a self-adaptive algorithm.[39]

Layer 3: The links of the nodes in this layer perform precondition matching of fuzzy-logic rules. The node in this layer performs a fuzzy AND operation on its incoming links.

Layer 4: This layer is composed of fuzzy sets representing its action responses. Similarly, the number of fuzzy sets belonging to an action response is also determined by either prior experience or a self-adaptive algorithm. The links in this layer perform postcondition matching of fuzzy-logic rules. The nodes in this layer perform a fuzzy OR operation on its incoming links.

Layer 5: There are two kinds of nodes in this layer. First kind of nodes (with down-up arrow) are the output of defuzzifiers that defuzzify the fuzzy sets in layer 4. Another kind of nodes (with up-down arrow) provide the channel to fuzzify each action response from task examples for supervised learning of the neuro-fuzzy network. When the training is done, these nodes will be removed from the network.

When a skill is represented by a neuro-fuzzy network, a skill is learned by training the neuro-fuzzy network from task examples. The training process is mainly divided into four steps: (1) Partitioning the perceptual-stimulus space and the action–response space from task examples; (2) construction of an initial network structure; (3) optimization of the network structure; and (4) optimization of the network parameters. From steps (1) to (3), a self-organized learning scheme is developed to construct a neuro-fuzzy network structure. Step (1) initializes membership functions of fuzzy sets in layers 2 and 4 of the network; step (2) matches the precondition in layer 2 and postcondition in layer 4; step (3) eliminates and combines some of the initial fuzzy rules to optimize the structure; and step (4), which is implemented as a supervised learning scheme, is used to optimally adjust the parameters of the membership functions for desired outputs. The details can be found in ref. [38].

After a neuro-fuzzy network is trained by task examples via supervised learning, the SRUs represented by fuzzy rules are self-categorized from the neuro-fuzzy network. Figure 3 shows a SRU from a learned neuro-fuzzy network. It also shows that a SRU physically means a mapping from the domain of perceptual stimuli to the domain of action responses and is mathematically modeled as a fuzzy rule. SRUs do not provide timing information; instead, they are reactive based on the activation of stimuli. Skill outputs are generated by a network in Fig. 2 constructed by SRUs, but there is no conflict among SRUs' outputs because each SRU is activated by a near exclusive range in the input space of stimuli.

3.2. Resolution control

After initial SRUs have been self-categorized by supervised learning, resolution control is employed to prune unnecessary SRUs, resulting in a small number of significant SRUs. The idea of resolution control is illustrated in Figs. 4(a)–(d). Figures 4(a) and (b) show four SRUs A, B, C, and D, and each of these has two perceptual stimuli, S_1 and S_2, and two

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

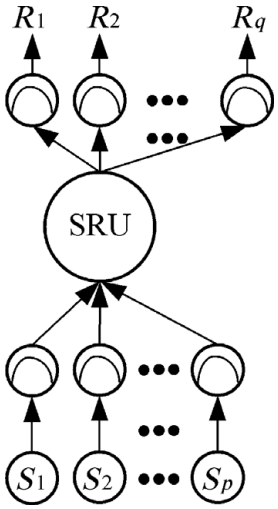

Fig. 3. Representation of a SRU extracted from a neuro-fuzzy network (S: perceptual stimulus; SRU: stimulus to response unit; R: action response).

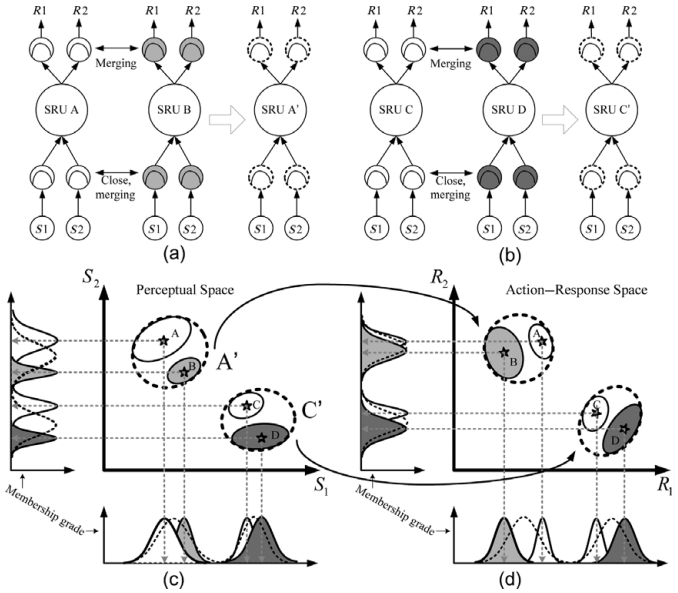

Fig. 4. Concept of resolution reduction of SRUs. (a) and (b) Six SRUs, A, B, C, D, A', and C', each of which has two perceptual stimuli, S_1 and S_2, and two action responses, R_1 and R_2. (c) Merging membership functions of perceptual stimuli of SRUs A and B, and C and D. (d) Merging membership functions of action responses of SRUs A and B, and C and D.

corresponding action responses, R_1 and R_2. In Fig. 4(a), SRUs A and B are close by a distance measure in the perceptual space, and so are SRUs C and D in Fig. 4(b). By resolution control, SRUs A and B, and C and D can be combined into SRUs A' and C' by merging their respective fuzzy sets.[40] The mean and variance of the merged fuzzy set A' from fuzzy sets A and B are expressed as

$$m_{A'} = \frac{m_A + m_B}{2},$$
$$\sigma_{A'} = \frac{|m_A - m_B|}{2} + \max(\sigma_A, \sigma_B), \qquad (1)$$

where (m_A, σ_A), (m_B, σ_B), and $(m_{A'}, \sigma_{A'})$ are the means and variances of fuzzy sets A, B, and A', respectively. The membership function of fuzzy set A' in Eq. (1) is bell-shaped because its mean and variance will be adjusted in the neuro-fuzzy network in the reinforcement-learning stage of skill decomposition.

From Fig. 4(c), A' (in dash line) covers a larger domain in the perceptual space, but A' loses some detailed perceptual information that has been described by A and B; so does C'. Thus, the resolution of perceptual stimuli of A' is lower than the original resolution provided by A and B, and so is the resolution of SRU C'. Meanwhile, extraneous actions generated from unnecessary SRUs are averaged out with other significant SRUs by the process of resolution control.

In our proposed framework, we construct a resolution binary tree (RBT) in order to generate a small number of significant SRUs. The concept of RBT is to establish a lookup table that describes means and variances of the membership functions of the fuzzy sets belonging to perceptual stimuli and action responses for each significant SRU. The advantage of this RBT is that the number of SRUs quickly decreases by half, and eventually to a small number of significant SRUs. The procedure of building a RBT is described by the RBT algorithm.

RBT algorithm: Given the initial SRUs from a learned neuro-fuzzy network, the RBT algorithm generates the reduced number of significant SRUs by grouping closest SRUs whose locations are measured in the perceptual-stimuli space. The RBT algorithm consists of two procedures – a top-down (steps **T1**–**T8**) and a bottom-up (steps **T9**–**T11**) procedures. In the top-down procedure, the algorithm recursively classifies each cluster into two sub-clusters until the number of SRUs in a sub-cluster is one. If the number of initial SRUs is odd, then one cluster has one more SRU than the other. The clustering algorithm is implemented by a K-means method by measuring the Euclidean distance (distance between the means of the fuzzy sets belonging to the same perceptual stimulus of two SRUs), and it ends up with generating a binary tree where each leaf node represents a SRU. From steps **T1** to **T5**, K-means method is applied to clustering SRUs. Steps **T6** and **T7** keep the number of SRUs in cluster(2) greater or equal than the number of SRUs in cluster(1) by reassigning the closest SRUs in one cluster to the other cluster. In step **T8**, it repeats to split each node of the tree into two child nodes with an equal number of SRUs through steps **T1** and **T7** until the leave nodes of the tree are created with one or zero SRU.

After the binary tree of SRUs is constructed, step **T9** starts to build a new tree in which nodes are the candidates of significant SRUs at the bottom layer of the binary tree. Step **T10** merges the two SRUs from the child nodes into a new SRU in their parent node. If a node's neighboring node (defined as having the same parent node) at the bottom layer is a dummy node (defined as representing no SRU), there is no merging to be performed with its neighboring node. Step **T11** repeats step **T10** from the bottom to the top of the binary tree. Thus, in every layer of the tree, it denotes a reduced number of SRUs.

T1. [*Determination of two initial centers.*] Randomly choose SRU_i and SRU_j from SRU_1 to SRU_n as $cluster(1)_{center}$ and $cluster(2)_{center}$, respectively. Also,

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

$|cluster(1)| \leftarrow 0$ and $|cluster(2)| \leftarrow 0$, where $|\cdot|$ is the count of number of SRUs in the cluster.

T2. [*Cluster indication for SRUs.*] $c_{SRU_k} \leftarrow argmin_{w=\{1,2\}} \| SRU_k - cluster(w)_{center} \|$ and $|cluster(c_{SRU_k})| \leftarrow |cluster(c_{SRU_k})| + 1$ for $1 \leq k \leq n$, where c_{SRU_k} is the cluster of SRU_k and $\|\cdot\|$ is the Euclidean distance from SRU_i to SRU_j and is defined as $\sqrt{\sum_{k=1}^{p} (Mean_k^{SRU_i} - Mean_k^{SRU_j})^2}$, where p is the number of stimuli, and $Mean_k^{SRU_i}$ and $Mean_k^{SRU_j}$ denote the means of the fuzzy set of stimulus k belonging to SRU_i and SRU_j, respectively.

T3. [*Center update.*] Calculate $cluster(1)_{center}$ and $cluster(2)_{center}$ by averaging the means of stimuli of the SRUs in $cluster(1)$ and $cluster(2)$.

T4. [*Error calculation of cluster.*] $E \leftarrow \sum_{w=\{1,2\}} \sum \| SRU_k - cluster(w)_{center} \|$ $\forall SRU_k \in cluster(w)$.

T5. [*Check error convergence.*] IF $|E_{current} - E_{previous}| \leq \tau_1$ and $|E_{current}| \leq \tau_2$, where τ_1 and τ_2 are design thresholds, THEN continues, ELSE go to step **T2**.

T6. [*Size difference of cluster.*] $\Delta N \leftarrow |cluster(1)| - |cluster(2)|$.

T7. [*Cluster re-indication for SRUs.*] IF $\Delta N \geq 0$, THEN assign ceil$(\Delta N/2)$ number of *SRU* from $cluster(1)$ to $cluster(2)$, which is closest (the shortest Euclidean distance) to any $SRU \in cluster(2)$, ELSE assign floor$(\Delta N/2)$ number of *SRU* from $cluster(2)$ to $cluster(1)$, which is closest to any $SRU \in cluster(1)$.

T8. [*Repeat clustering.*] Designate $cluster(1)$ and $cluster(2)$ as parent nodes in the binary tree. Repeat steps **T1** to **T7** for each parent node, resulting in two child nodes until leave nodes are created where $|cluster(w)| = 1$ or 0 for $w = 1, 2$.

T9. [*Initialization for generating significant SRUs.*] Start at the bottom layer of the tree.

T10. [*Merging of child nodes.*] IF either of the two child nodes belonging to the same parent node is dummy ($|cluster(w)| = 0$), THEN assign the SRU in the non-dummy node as the new SRU in the parent node ELSE merge the two SRUs from the child nodes into the new SRU in the parent node.

T11. [*Repeat merging.*] Repeat step **T10** until the top layer of the tree is reached.

END RBT Algorithm.

Figure 5(a) shows the top-down procedure of building a RBT, and the number of initial SRUs is 15. Since 15 is odd, the number of one cluster is one more than the other, and they are 8 and 7. By repeating steps **T1**–**T8** of the RBT algorithm, the number of SRU is one down to the bottom. Also, there is a dummy node at the bottom. Since each node represents a SRU, the initial number of SRUs is between 2^p and 2^{p+1}, where p is a non-negative integer and the number at the bottom is 2^{p+1}, where $p = 3$ and $p + 2$ are the numbers of layers of the RBT. In Fig. 5(b), it illustrates how the RBT algorithm generates a reduced number of SRUs by merging

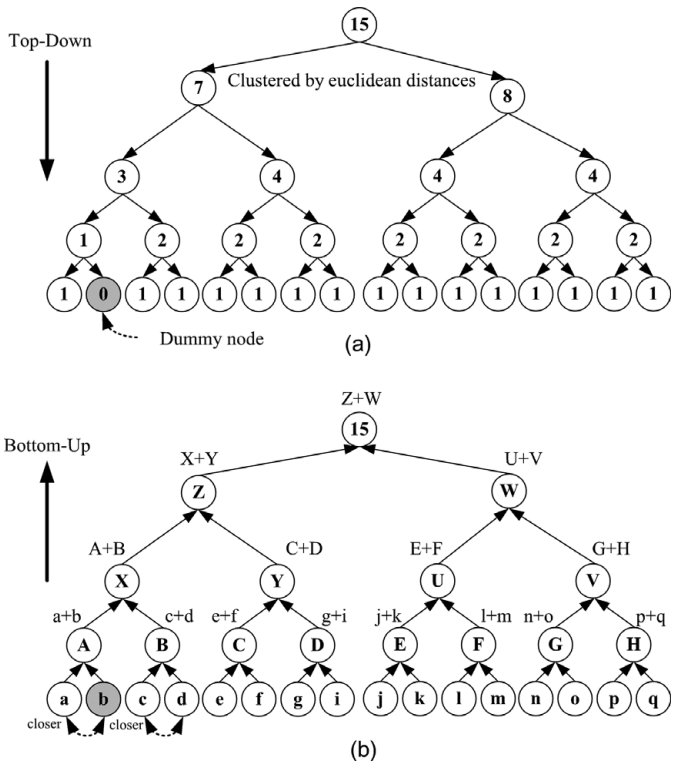

Fig. 5. (a) Top-down procedure. (b) Bottom-up procedure.

neighboring nodes from the bottom layer to the top layer of a RBT. When the layer is up by one, the number of SRUs decreases by half.

3.3. Reinforcement learning

After the supervised learning, the structure and parameters of the neuro-fuzzy network (see Fig. 2) have been learned, and the resolution control generates a small number of significant SRUs. In the third stage, reinforcement learning is introduced to test and validate whether these significant SRUs are sufficient for accomplishing the task. Thus, the reinforcement-learning stage has two main functions: providing a process for validating the suitability of small number of significant SRUs, and providing a learning mechanism for adjusting the parameters of these SRUs, resulting in the changed behavioral patterns of SRUs to accomplish the task.

In the reinforcement-learning stage, a skill is represented by significant SRUs, and an adaptive-heuristic-critic algorithm is adopted to adjust the behaviors of SRUs to achieve the goal of the given task. Figure 6 shows a neuro-fuzzy network with reinforcement learning. The neuro-fuzzy network organizes SRUs by using two major subnets, i.e., critic and action subnets.

The purpose of the critic and action subnets is to obtain appropriate behaviors of SRUs for accomplishing a given task. The SRUs in the action subnet provide well-structured, multi-step actions rather than single-step actions in a conventional reinforcement approach because a SRU is mathematically modeled as a fuzzy rule that can be used to generate actions for many states in the state space. On the other hand, the critic subnet provides the parameter learning for tuning behaviors of SRUs because a critic is

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

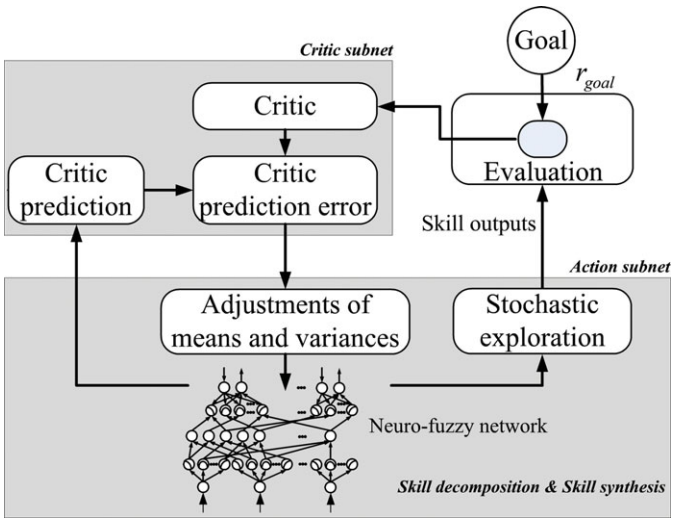

Fig. 6. (Colour online) A neuro-fuzzy network in the reinforcement-learning stage.

an indication showing the goal achievement for the task, and it strongly drives and shapes the learning process and the results of reinforcement learning. With well-structured actions and parameter learning, the skill-learning time in the reinforcement-learning stage is dramatically reduced. For the critic subnet, it consists of critic, critic prediction, and critic-prediction error. The critic is a binary signal providing success or failure for the outcome of the performance. The critic prediction, $p(t)$, is to predict the critic signal in order to choose a better action from the action subnet at time step $(t-1)$. In other words, the predicted critic can help the action subnet to perform a more efficient random search, since reinforcement learning is an exploitation-and-exploration process. The critic-prediction error is used to calculate the error between the predicted and actual critics, and update the parameters of critic prediction in the critic subnet and SRUs in the action subnet.

For the action subnet, it consists of SRUs, stochastic exploration, and mean-and-variance adjustments of fuzzy sets. SRUs function as initial action templates and are identified from task examples in the supervised-learning stage. The outputs of SRUs are wired together to produce the network's outputs. Thus, the network's outputs are generated on the basis of exploiting SRUs. In addition, stochastic exploration is added at the network's output to assist the reinforcement-learning stage in exploring a near optimal solution. Meanwhile, a critic-prediction error is fed back to adjust the behaviors of SRUs to generate a correct predicted critic through changing the means and variances of the membership functions of fuzzy sets. These changes in membership functions are performed by backpropagation of the critic signal until the task is achieved. In our mobile-robot experiments, we set 100 trials as the upper limit of the number of trials to judge whether the robot achieves the given task or not. This upper limit can be adjusted – depending on how quickly we expect the robot to accomplish a task. Although a small upper limit can be used to quickly judge whether a robot succeeds or fails a task, it may cause the robot to easily fail the task. Thus, an adequate upper limit of the number of trials

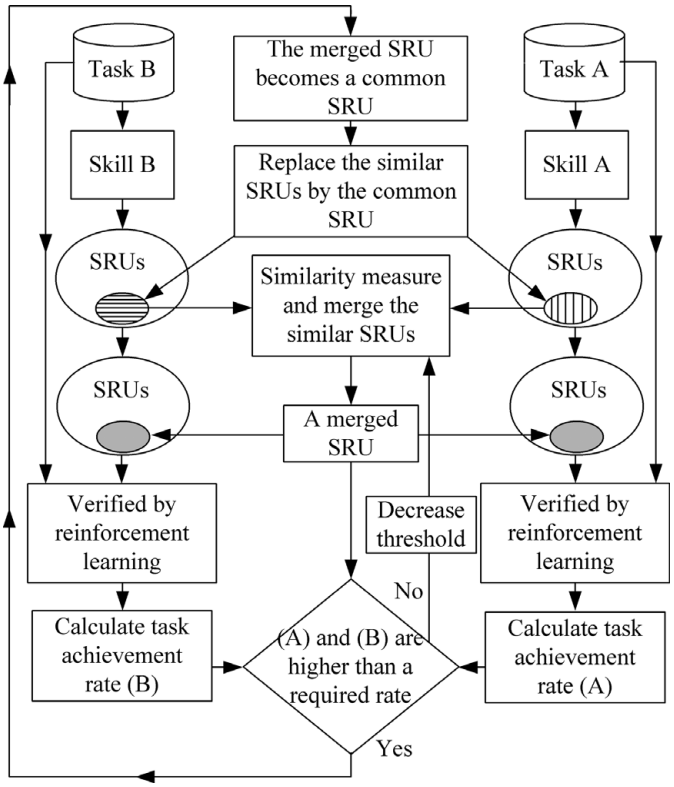

Fig. 7. Proposed approach of determining common SRUs between two skills A and B.

is necessary. The updating rules and detailed calculations of the adaptive-heuristic-critic algorithm can be found in refs. [38, 41] .

4. Skill Synthesis Phase

Skill synthesis targets learning new skills for a new task by self-organizing learned skills. We employ a two-stage procedure to self-organize learned skills into a new skill – determining common SRUs and self-organizing new skills from learned skills.

4.1. Determining common SRUs

The purpose of determining common SRUs between similar skills is to keep the pool of SRUs to a minimum. When a robot has learned skills for similar tasks, it is desirable to find their common SRUs. Hence, the robot can use less resources for storing similar skills. Determining common SRUs between two similar skills is to identify the maximum number of common SRUs that can be used in achieving their tasks by measuring the "similarity" between SRUs. Since SRUs are represented by fuzzy rules, similarity between SRUs can be measured by a similarity measure of fuzzy rules.[40,42] Figure 7 shows the flow chart of the proposed approach that identifies the two most similar SRUs between two skills by setting a high threshold value for the similarity measure of SRUs, merges them into a merged SRU by statistically averaging their membership functions, replaces these two similar SRUs with the merged one, and then verifies the two skills with the merged SRU by reinforcement learning. The reinforcement learning, using the same neuro-fuzzy network structure in the

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

skill decomposition, verifies whether the two skills with the merged SRU can succeed the two original tasks as measured by a task achievement rate. If it is successful, then the merged SRU becomes a common SRU for the two skills. The approach will then repeat the above procedure and find the next two most similar SRUs by lowering the threshold value of the similarity measure of the two SRUs until the maximum number of common SRUs can be found. A task achievement rate is defined as the ratio of the number of successful tests to the number of total tests, where a successful test is defined as when the robot can accomplish a given task within an acceptable number of trials. We selected 100 tests as the number of total tests for the simulations and 20 tests for the mobile robot experiments, and similarly we selected 100 trials as the acceptable number of trials for the simulations and 20 trials for the mobile robot experiments. It is possible that a more complex task may need more trials.

In order to develop a similarity measure between two SRUs, we employ the similarity of rule premise (SRP) and similarity of rule consequent (SRC)[43] to calculate this measure. For example, two fuzzy rules, $Rule^i$ and $Rule^j$, are represented as

$Rule^i$: IF(S_1 is T^i_{s1} and $\cdots$ and S_k is T^i_{sk} and $\cdots$ and S_p is T^i_{sp}), THEN (R_1 is T^i_{r1} and $\cdots$ and R_k is T^i_{rk} and $\cdots$ and R_q is ${}^iT_{rq}$), and $Rule^j$: IF(S_1 is T^j_{s1} and $\cdots$ and S_k is T^j_{sk} and $\cdots$ and S_p is T^j_{sp}), THEN (R_1 is T^j_{r1} and $\cdots$ and R_k is T^j_{rk} and $\cdots$ and R_q is ${}^jT_{rq}$), where T^i_{sk} and T^j_{sk} are fuzzy sets of S_k of a perceptual stimulus, T^i_{rk} and T^j_{rk} are fuzzy sets of R_k of an action response, and p and q are the numbers of perceptual stimuli and action responses, respectively. The SRP and SRC between these two fuzzy rules are respectively defined as

$$SRP(i, j) = \min_{k=1}^{p} S_{set}\left(T^i_{sk}, T^j_{sk}\right), \tag{2}$$

and

$$SRC(i, j) = \min_{k=1}^{q} S_{set}\left(R^i_{rk}, R^j_{rk}\right), \tag{3}$$

where $SRP(i, j)$ is the SRP between $Rule^i$ and $Rule^j$, $SRC(i, j)$ is the SRC between $Rule^i$ and $Rule^j$, and $S_{set}(T^i_{sk}, T^j_{sk})$ is the similarity measure between two fuzzy sets T^i_{sk} and T^j_{sk}. In Eqs. (2) and (3), assuming the membership functions of all fuzzy sets are bell-shaped, the similarity measure between two fuzzy sets is defined as

$$S_{set}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{\sigma_A\sqrt{\pi} + \sigma_B\sqrt{\pi} - |A \cap B|}, \tag{4}$$

where $|A \cap B|$ and $|A \cup B|$ are the cardinality of intersection and union of fuzzy sets A and B, respectively, and $|A \cap B|$ is calculated as in ref. [38]. The areas of the bell-shaped function of the fuzzy sets A and B are $\sigma_A\sqrt{\pi}$ and $\sigma_B\sqrt{\pi}$, respectively.[44] Then, the similarity measure of two fuzzy rules is defined as

$$S_{rule}(i, j) = \min(SRP(i, j), SRC(i, j)), \tag{5}$$

where $S_{rule}(i, j)$, $0 \le S_{rule}(i, j) \le 1$, is the similarity measure between two fuzzy rules, $Rule^i$ and $Rule^j$. In Eq. (5), the degree of equality of two fuzzy rules is defined by taking the minimum operation of SRP and SRC. Thus, fewer pairs of similar SRUs can be found by having a high threshold value of S_{rule}. On the contrary, more pairs of similar SRUs can be found when the threshold value of S_{rule} is lowered.

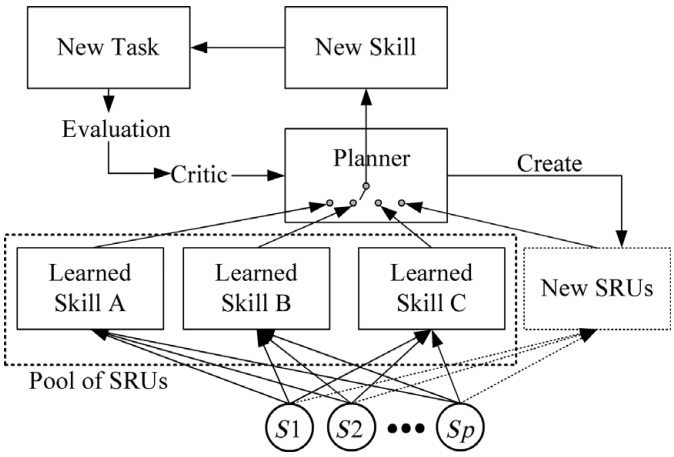

Fig. 8. Illustration of self-organizing a new skill for a new task.

4.2. Self-organizing new skills with reinforcement learning

In skill synthesis, a new skill is learned by acquired skills composed of SRUs. However, it is possible that the robot may have difficulties in accomplishing a new task in some situations only using the learned skills. Hence, synthesizing a new skill from the learned skills may not be able to complete the new task, and additional new SRUs that are different from the learned skills must be learned to assist the robot in synthesizing the new skill. Thus, learned skills and additional new SRUs will be self-organized into a new skill for a new task. Once a new skill is self-organized, the reinforcement learning with a Q-learning algorithm will be utilized to validate whether the self-organized new skill can be used to accomplish the new task or not. Figure 8 shows the proposed skill-synthesis framework that integrates the high-level planning and low-level implementation from perceptual stimuli to action responses by SRUs. The critic in Fig. 8 shows whether the new skill fails the task or not, and it also guides the planner how to determine the sequence of learned skills for execution and when to create new SRUs for accomplishing the task. This integration of high-level planning and low-level implementation is a salient feature of the proposed skill-synthesis framework.

In order to save the learning effort by planning a sequence of learned skills for execution, we propose a self-organizing-skill (SOS) algorithm to sequentially plan learned skills composed of SRUs and create new SRUs, if necessary, into a new skill, and then validate the self-organized new skill by the reinforcement learning with a Q-learning algorithm.

Figure 9 illustrates how SOS algorithm works. The task is to ask a robot to go from the start state (leftmost black grid) to the end state (rightmost black grid). Each grid in Fig. 9 denotes the resolution of the input space of perceptual stimuli. Different learned skills composed of SRUs can be performed on each grid that is in the set of possible initial perceptual states of these learned skills. A middle black grid denotes the

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

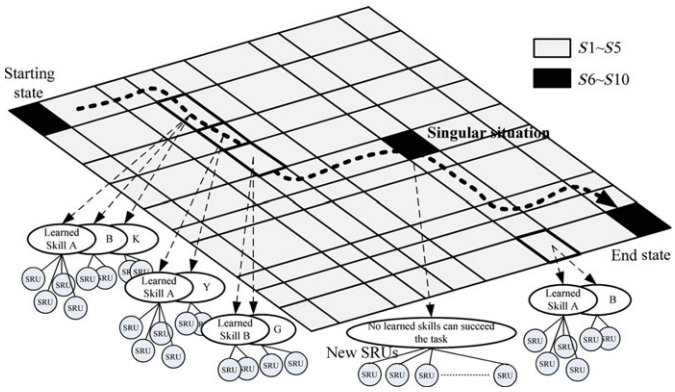

Fig. 9. (Colour online) Illustration of the proposed SOS algorithm. Leftmost black grid: starting state; rightmost black grid: end state; middle black grid: singular situation.

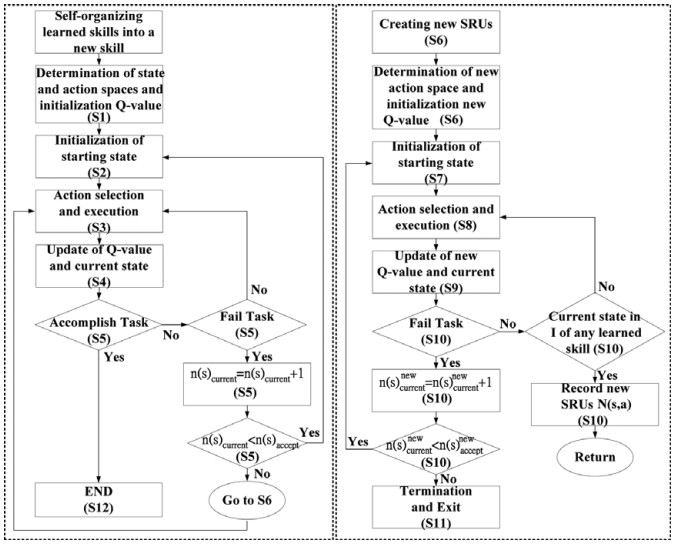

Fig. 10. Flow chart of the proposed self-organizing-skill (SOS) algorithm.

situation that the robot always fails the task using all existing learned skills more than an acceptable number of trials. SOS algorithm applies steps **S1**–**S5** and **S6**–**S10** to the robot when it travels to the gray (normal situations) and middle black (singular situations) grids, respectively. A singular situation s is detected by step **S5** when $n(s)_{current} > n(s)_{accept}$, where $n(s)_{current}$ is the current number of trials at the state s, and n_{accept} is an acceptable number of trials for steps **S1**–**S5**. Eventually, the robot can find a path (dashed trajectory) to go from the start state to the end state.

SOS algorithm: Given skills acquired by the proposed skill-decomposition framework, the SOS algorithm sequentially plans them and creates new SRUs for accomplishing a new task. It is implemented by two nested Q-learning loops (steps **S1**–**S5** and **S6**–**S10**). Figure 10 shows the flow chart of the proposed SOS algorithm. The left box, including steps **S1**–**S5**, is the outer loop for determining a sequence of learned skills for execution; the right box, including steps **S6**–**S10**, is the inner loop for creating new SRUs. Steps **S1**–**S4** initialize and update the Q-value. Step **S5** checks whether the robot succeeds the task within an acceptable number of trials. If not, the algorithm proceeds to step **S6** to create new SRUs to resolve the task difficulties that cannot be done by the existing learned skills. New SRU candidates are created with different actions by uniformly allocating their fuzzy sets of action responses. Step **S8** chooses a new SRU candidate using the best Q-value and executes it to resolve the difficult situation of the task, which is similar to the task done by step **S3** in the outer loop. Step **S10** evaluates whether a robot passes the difficult condition within another acceptable number of trials. If yes, new SRUs are denoted as $N(s, a)$. Afterwards, the outer loop takes over the flow control of the SOS algorithm and continues to explore the following sequence of the learned skills for execution until the robot completes the task.

S1. [*Determination of state and action spaces, S and M, and initialization of Q-values for learned skills.*] $S \leftarrow$ the perceptual state space of SRUs; $M \leftarrow$ the set of learned skills. Initialize Q values, $Q(s, m)$, with a small positive numbers, where $s \in S$ and $m \in M$.

S2. [*Initialization of state.*] $s \leftarrow$ the initial state of the task. It is an element in the set of possible initial perceptual states of the learned skill, I.

S3. [*Selection and execution of a learned skill.*] $m \leftarrow$ choose a learned skill from M according to s by using the policy derived from Q, where s is in the I of m. Then, take action of m, observe next state s', and get reward r, where s' is the state in which another learned skill m' in M is taken.

S4. [*Update Q-value and current state.*] $Q(s, m) \leftarrow (1 - \alpha)Q(s, m) + \alpha[r + \gamma max_{m'} Q(s', m') - Q(s, m)]$, where m' is the learned skill associated with s', α is a learning-rate parameter, and γ is a discount-rate parameter; then set $s \leftarrow s'$.

S5. [*Evaluation.*] `IF` s is the state of accomplishing the task, `THEN` go to **S12**, `ELSEIF` s does not fail the task, `THEN` go to **S3**, `ELSEIF` $(n(s)_{current} \leftarrow n(s)_{current} + 1) < n_{accept}$, `THEN` go to **S2**, `ELSE` $s_{new} \leftarrow s$ and go to **S6** for creating new SRUs.

S6. [*Determination of action space of new SRU candidates and initialization of Q-values of new SRU candidates.*] Given an adjustable number N_i for each action response, $1 \leq i \leq q$, where q is the number of action responses. Partition each action response into N_i equal parts and select the center of each partition as the mean of the fuzzy set of an action response belonging to a new SRU candidate, a_j. The variance of the new SRU candidate is $(m_i - m_k)/2$, where m_k is the nearest mean of another fuzzy set belonging to the same action response. The combination of all action responses partitioned from q action responses results in different new SRU candidates, a_j, and the action space of the new SRU candidates for the task, $A_{new} \leftarrow \{a_j\}_{j=1}^{\prod_{i=1}^{i=q} N_i}$. After the action space of new SRU candidates is determined, initialize $Q_{new}(s, a)$ arbitrarily, where $s \in S$ and $a \in A_{new}$.

S7. [*Initialization of state.*] $s \leftarrow s_{new}$.

S8. [*Selection and execution of new SRU candidates.*] $a \leftarrow$ choose an action from A_{new} according to s by using the policy derived from Q_{new}. Then, take action of a, observe next state s', and get reward r, where s' is the

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

next state in which another new SRU, or a SRU of a learned skill is taken.

S9. [*Update Q-value and current state of new SRU candidates.*] $Q_{new}(s,a) \leftarrow (1-\beta)Q_{new}(s,a) + \beta[r + \delta \max_{a'} Q_{new}(s',a') - Q_{new}(s,a)]$, where a' is the action taken in s', β is a learning-rate parameter, and δ is a discount-rate parameter; then set $s \leftarrow s'$.

S10. [*Evaluation of new SRU candidates.*] `IF` s does not fail the task and s is in the I of a learned skill, `THEN` record the actions from s_{new} to s as the set of new SRUs, $N(s,a)$ and go to **S3**, `ELSEIF` s does not fail the task and s is not in the I of a learned skill, `THEN` go to **S8**, `ELSEIF` $(n(s)^{new}_{current} \leftarrow n(s)^{new}_{current} + 1) < n^{new}_{accept}$, `THEN` go to **S7**, `ELSE` go to **S11**, where $n(s)^{new}_{current}$ is the current number of trials for creating new SRUs at the state s, and n^{new}_{accept} is an acceptable number of trials for creating new SRUs.

S11. [*Termination of self-organizing a new skill.*] A new skill cannot be self-organized by the created SRU candidates. Exit.

S12. [*End of self-organizing a new skill.*] The policy of $Q(s,m)$ is the new skill, and $N(s,a)$ is the set of new SRUs for the task. Exit.

END SOS Algorithm.

5. Computer Simulations and Experimental Work

In the simulations and experiments, we validated the proposed framework by some traveling skills of mobile robot. Although some previous work had good results in mobile robot navigation,[45,46] the purpose of the experiments focused on how to learn a new skill from the obtained simple skills. Computer simulations and experiments using the player/stage mobile robot control software and an ActivMedia Pioneer 3-DX mobile robot were conducted to validate the performance of the proposed neuro-fuzzy-based, self-organizing skill-learning framework.

5.1. Experiment 1: Hallway-passing skill

In this example, we implemented a simple rule for the robot to pass a hallway – when the robot approaches a wall, it will turn toward the other side to avoid bumping into the wall. If a neuro-fuzzy network with supervised learning is utilized to learn the action patterns of the autonomous robot, it will result in a zig-zag fashion of passing the hallway because the neuro-fuzzy network will learn the trajectory faithfully. However, passing a hallway in a zig-zag fashion is not the skill that we would like the robot to learn; thus, we employed the proposed three-stage skill decomposition to learn the hallway-passing skill utilizing SRUs and discover a sufficient number of SRUs to pass the hallway without moving in a zig-zag fashion.

We first used a five-layer, neuro-fuzzy network with supervised learning to capture the action patterns from task examples and categorize them into initial SRUs. Figure 11(a) shows a task example of passing the hallway utilizing the simple rule. The training data were collected by a SICK LMS-200 laser ranger equipped on the Pioneer P3-DX mobile robot. The hallway in our simulation is a hallway of the ground floor of the EE building at Purdue University. The

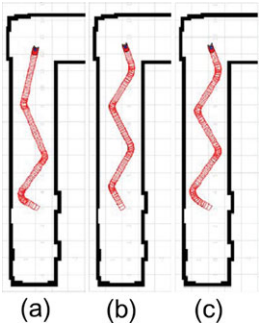

Fig. 11. (Colour online) Skill of passing a hallway. (a) Task example. (b) and (c) Training results from the neuro-fuzzy network with an initial robot orientation of 30° and 60°, respectively, with respect to the upright.

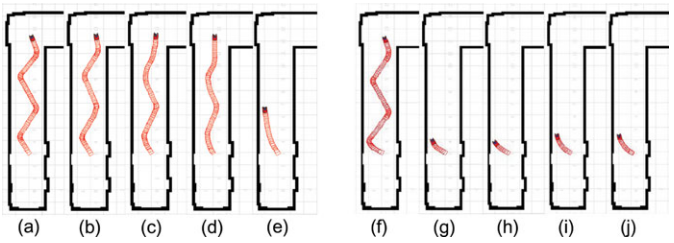

Fig. 12. (Colour online) Skill of passing a hallway with different numbers of SRUs when the initial angle of robot orientation is 30°: (a) 128; (b) 64; (c) 32; (d) 16; (e) 8 SRUs. Similarly, when the initial angle of robot orientation is 60°: (f) 128; (g) 64; (h) 32; (i) 16; (j) 8 SRUs.

training data for the neuro-fuzzy network comprised four sensory inputs: minimum distance from the right-hand wall, minimum distance from the left-hand wall, the center-most distance from obstacles, and its current orientation, and two actuator outputs: turning angle and speed. The sampling rate was 10 Hz. We set the number of membership functions to 5 for each perceptual stimulus and 10 for each action response to generate 124 initial SRUs. Figures 11(b) and (c) show the training results from the neuro-fuzzy network.

With the initial 124 SRUs, resolution control constructed a RBT. At the bottom layer of the tree, there are 128 nodes (SRUs) with 4 "dummy nodes" because $2^6 < 124 < 2^7$. After resolution control, Figs. 12(a)–(e) show the simulation results of different resolutions of SRUs when the angle of initial robot orientation was 30° . In Fig. 12(a), when the resolution is 128 (the highest), the result was similar to Fig. 11(b). Other resolutions of SRUs are shown in Figs. 12(b)–(e). When the number of SRUs was reduced to less than 128, the mobile robot could pass the hallway with less zig-zag actions. It came out with a more straight-line hallway passing skill to achieve the goal. However, Fig. 12(e) shows that when the number of SRUs was 8, the robot could not pass the hallway in the first trial. Also, Figs. 12(g)–(j) show that when the number of SRUs was less than 128 and the initial angle of robot orientation was 60° , the robot still could not pass the hallway in the first trial. With these failures, the reinforcement learning was used to adjust the behaviors from these small numbers of SRUs.

The adjustment process tuned the behaviors of the hallway-passing skill on the basis of SRUs. Figure 6 shows how the

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

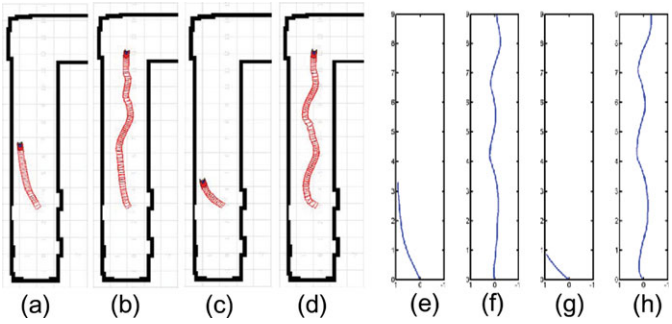

Fig. 13. (Colour online) The adjustment of the behaviors of 8 SRUs. When the initial angle of robot orientation is 30°, results from computer simulations: (a) Trial 1 before the reinforcement learning, (b) Trial 2 after the reinforcement learning; experimental results: (e) Trial 1, (f) Trial 2. Similarly, when the initial angle of robot orientation is 60°, simulation results: (c) Trial 1, (d) Trial 3; experimental results: (g) Trial 1, (h) Trial 3.

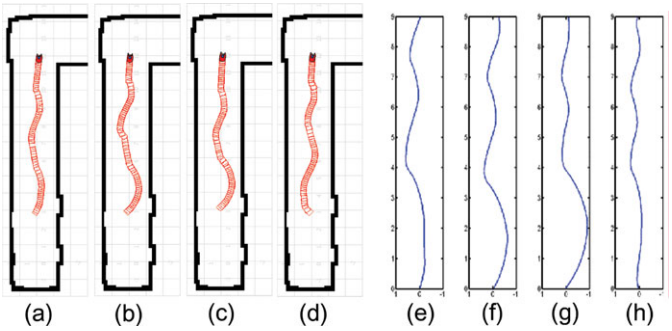

Fig. 14. (Colour online) (a)–(d) and (e)–(h) are simulation and experimental results, respectively, with 8 SRUs when the initial angle of robot orientation is: −30° ((a) & (e)), −45°((b) & (f)), −60°((c) & (g)), and 45° ((d) & (h)).

critic r adjusts the means and variances of the fuzzy sets belonging to perceptual stimuli and action responses. The critic is given 0 at each time step when the robot does not fail the task, or −1 when it fails the task. Figures 13(a)–(b) and (e)–(f) show the reinforcement learning that took two trials to complete the task by adjusting the action outputs of the SRUs when the number of SRUs was 8 and the initial angle of robot orientation was 30°. It shows that the hallway-passing task was accomplished using the fuzzy rule system (SRUs) that was adjusted a bit in two trials. When the initial angle of robot orientation was 60°, Figs. 13(c)–(d) and (g)–(h) show the reinforcement learning that took 3 trials to complete the task. In 100 of continuous tests with $\alpha_m^r = 0.3$, $\alpha_\sigma^r = 0.3$, $\alpha_m^s = 0.3$, $\alpha_\sigma^s = 0.3$, $\omega = 0.1$, $\gamma = 0.95$, $\beta = 0.1$, and $\lambda = 0.8$, it took 8.21 and 12.12 trials on the average and 1.31 and 1.73 trials on the variance for 8 SRUs in a test to finish the task when the initial angle of robot orientation was 30° and 60°, respectively. Other testing results for −30°, −45°, −60°, and 45° of robot orientation are shown in Fig. 14. Table I shows the parameters of the fuzzy rule of each SRU whose structure is shown Fig. 3. In Table I, the mean and variance have been normalized by real data and the mean of action response 1 in the parenthesis indicates the turn rate of the robot.

In this example, we set the upper limit of 100 trials in the reinforcement learning for distinguishing whether the

Table I. Eight corresponding fuzzy rules to 8 SRUs (m: mean and σ: variance).

	Input stimuli								Action responses			
	Perceptual Stimulus 1		Perceptual Stimulus 2		Perceptual Stimulus 3		Perceptual Stimulus 4		Action Response 1		Action Response 2	
	m_1	σ_1	m_2	σ_2	m_3	σ_3	m_4	σ_4	m_1	σ_1	m_2	σ_2
SRU0	.307,135	.162,940	.323,917	.197,859	.117,623	.080,543	.897,270	.059,210	.563,663 (.002)	.155,065	.435,634	.339,545
SRU1	.606,163	.121,902	.227,496	.201,534	.334,590	.230,338	.878,978	.073,309	.459,994 (–0.094)	.119,415	.427,282	.332,554
SRU2	.087,823	.076,026	.865,135	.116,821	.355,033	.184,114	.887,569	.076,698	.692,344 (0.162)	.111,096	.218,176	.318,763
SRU3	.197,088	.143,567	.691,745	.172,283	.116,580	.072,050	.892,974	.057,909	.608,686 (0.07)	.102,001	.467,580	.316,834
SRU4	.153,889	.101,069	.828,447	.128,712	.301,662	.153,598	.159,073	.234,007	.606,190 (0.067)	.104,941	.431,794	.286,574
SRU5	.271,923	.199,764	.515,920	.171,400	.135,641	.091,333	.766,523	.055,985	.580,554 (0.039)	.082,078	.552,645	.267,480
SRU6	.394,751	.221,680	.172,122	.145,854	.089,808	.069,726	.076,109	.043,237	.143,257 (–0.442)	.163,482	.021,121	.064,024
SRU7	.470,241	.154,695	.489,109	.202,674	.204,562	.144,388	.077,460	.048,898	.407,907 (–0.151)	.010,011	.906,588	.544,357

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

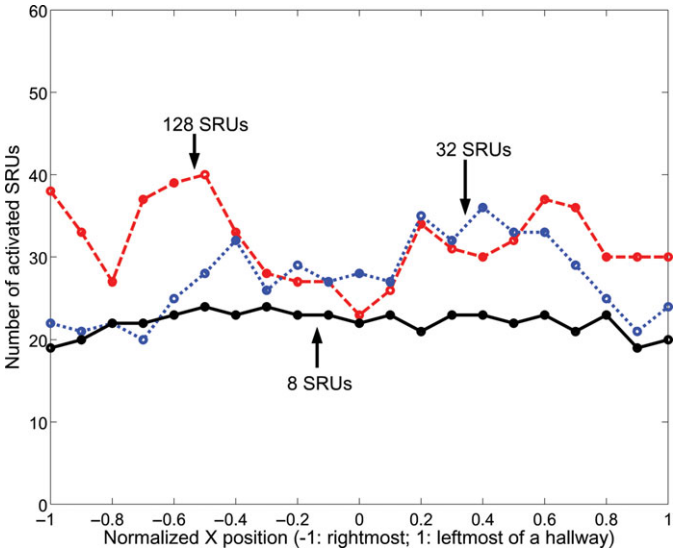

Fig. 15. (Colour online) Number of activated SRUs by summing up forward orientations (dashed line: 128; dotted line: 32; solid line: 8 SRUs). The number of activated SRUs is not highest at ± 1 since the zig-zag motion (extraneous motion) is removed by resolution control.

resolution of SRUs is successful or not in completing the task. If the number of trials is under 100 and the task is completed, then the resolution of SRUs succeeds the task; otherwise it fails the task. As for using 4 or 2 SRUs, both could not pass the hallway within the limit of 100 trials. Thus, 4 or 2 SRUs are not sufficient for accomplishing the task. From the results of this example, having 8 SRUs is sufficient and appropriate for the robot to learn the skill of passing the hallway.

Since having 8 SRUs is sufficient for accomplishing the hallway passing task, we can interpret these 8 SRUs as fuzzy rules that can be extracted from the final neuro-fuzzy network. Figure 15 shows the number of SRUs activated at a specific position with a 10^o increment between -90^o and 90^o, where the x-axis represents the normalized position (from -1 to 1: from right to left) of the robot in the hallway. In Fig. 15, when the number of SRUs is 128, there are more SRUs activated near the walls (the normalized x position is between -0.4 and -0.7, and 0.2 and 0.7) than the middle of the hallway (the normalized x position is between -0.3 and 0.1). After resolution control, 128 SRUs are reduced to 32 and 8 SRUs. Figure 15 also shows the number of activated SRUs with 32 and 8 SRUs. Apparently, the activated number of SRUs near the wall dropped dramatically when 128 SRUs were reduced to 8 SRUs. This result indicated that resolution control pruned unnecessary SRUs that induced extraneous actions moving near the walls. When the number of SRUs was pruned to 8, Fig. 15 shows its uniform-like distribution of activated number of SRUs for accomplishing the task.

5.2. Experiment 2: Determining common SRUs between the skills of traveling around an ellipse and a circle

In the second experiment, we want to identify the existence of common SRUs between two similar skills. We first manually controlled the P3-DX robot to perform two different skills, separately traveling around an ellipse and a circle in the simulation using player/stage. The training data from the simulation were obtained by recording the sensor inputs and actuator outputs (same as those in experiment 1) and shown in Figs. 16(a) and (b). Then, the robot acquired these two skills and identified 16 and 8 significant SRUs for traveling around the ellipse and the circle, respectively, by our proposed skill decomposition. These two skills were demonstrated in the simulation shown in Figs. 16(c) and (d) by using their respective numbers of significant SRUs. In addition, these two numbers of significant SRUs were tested for different sizes of ellipse and circle in the simulation as well as on the P3-DX mobile robot. Figures 16(e)–(f) and (g)–(h) show their simulation and experimental results of traveling around a smaller ellipse and circle, respectively. These results show that these two skills learned by their significant SRUs are general to a different size of ellipse and circle, respectively.

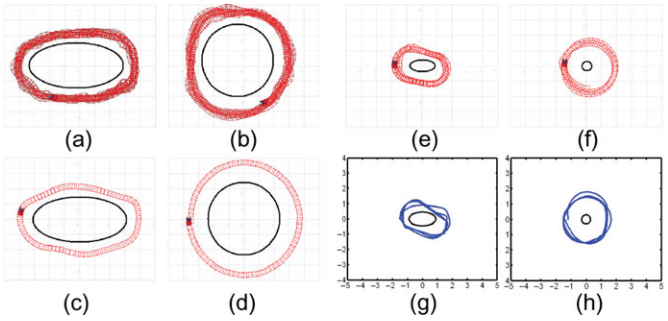

Fig. 16. (Colour online) (a) and (b): Training data of traveling around an ellipse and a circle, respectively. (c) Traveling around an ellipse by utilizing 16 sufficient SRUs. (d) Traveling around a circle by utilizing 8 sufficient SRUs. (e)–(f) and (g)–(h): Simulation and experimental results of traveling around an ellipse and a circle, respectively. (e) and (g): Utilizing 16 sufficient SRUs to travel another small ellipse. (f) and (h): Utilizing 8 sufficient SRUs to travel another small circle.

We applied a similarity measure to these two different sets of SRUs of the two skills to determine their common SRUs. The threshold value of the similarity measure was given by S_{rule} as defined in Eq. (5). Initially, we set a high threshold value of S_{rule} to determine a fewer number of common SRUs between them. Then, we decreased S_{rule} to determine more common SRUs. These two skills have no common SRU when $S_{rule} > 0.4$. However, when S_{rule} was decreased from 0.4 to 0.38, 0.27, and 0.05, the number of common SRUs was increased from 0 to 1, 4, and 8, respectively. We verified these numbers of common SRUs in the simulation and on the P3-DX mobile robot, and they all had a task achievement rate of one after the reinforcement learning. A task achievement rate has been defined in Section 3. Thus, it shows that the robot can perform each of these two skills to achieve their tasks by sharing these 8 common SRUs. Figure 17 shows the simulation and experimental results by utilizing these 8 common SRUs and 8 additional SRUs (i.e., 16 SRUs) for traveling around an ellipse, and using these 8 common SRUs for traveling around a circle.

The 8 common SRUs are the shared fuzzy rules for accomplishing both skills of traveling around an ellipse and a circle. Since a neuro-fuzzy network is isomorphic to a fuzzy logic control system,[38] we can easily interpret these 8 fuzzy rules from the trained neuro-fuzzy network as follows:

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

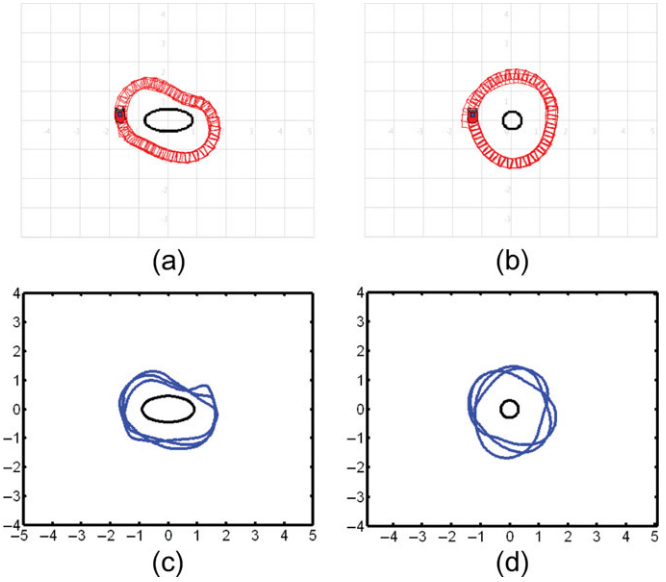

Fig. 17. (Colour online) Results of traveling around an ellipse and a circle with the 8 common SRUs, respectively. (a) and (b): Simulation. (c) and (d): Experiment.

SRU1): IF the robot is heading "very fairly" close to the ellipse or circle and its orientation is around 150°, THEN it will turn left "fairly" (orientation was 0°when the robot headed upright and counted clockwise); SRU2): IF the robot is heading "slightly" far from the ellipse or circle and its orientation is around 170°, THEN it will turn right "fairly"; SRU3): IF the robot is heading "fairly" close to the ellipse or circle and its orientation is around 170°, THEN it will turn right "very slightly"; SRU4): IF the robot is heading "slightly" far from the ellipse or circle and its orientation is around 60°, THEN it will turn right "slightly"; SRU5): IF the robot is heading "slightly" close to the ellipse or circle and its orientation is around 90°, THEN it will turn right "slightly"; SRU6): IF the robot is heading "straight" and its orientation is around 280°, THEN it will turn right "slightly"; SRU7): IF the robot is heading "fairly" far from the ellipse or circle and its orientation is around 60°, THEN it will turn right "fairly"; SRU8): IF the robot is heading "fairly" far from the ellipse or circle and its orientation is around 320°, THEN it will turn right "fairly". One of the salient features of using a neuro-fuzzy network to represent a learned skill is its ability to extract the fuzzy rules from the trained neuro-fuzzy network.

5.3. Experiment 3: Hallway-passing skill to avoid a circle-like obstacle

In this experiment, we want the robot to self-organize a new skill by utilizing learned skills from previous experiments. Here the new skill is to pass a hallway with a circle-like obstacle present, and the learned skills are the hallway-passing skill and the circle-traveling skill in the previous experiments. For both skills, they were represented by two different 8 significant SRUs identified in these experiments.

In order to self-organize a new skill from the learned skills in the proposed SOS algorithm, the state and action spaces of the new skill are represented and encoded by the significant SRUs of all the learned skills. After the state and action spaces

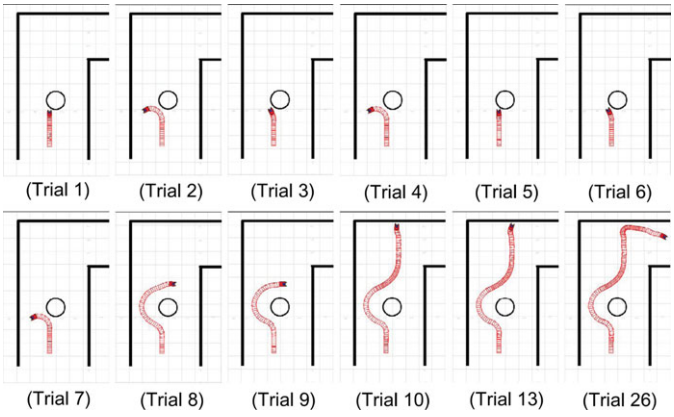

Fig. 18. (Colour online) Simulation results of the skill of passing a hallway when there is a circle-like obstacle and make a right turn.

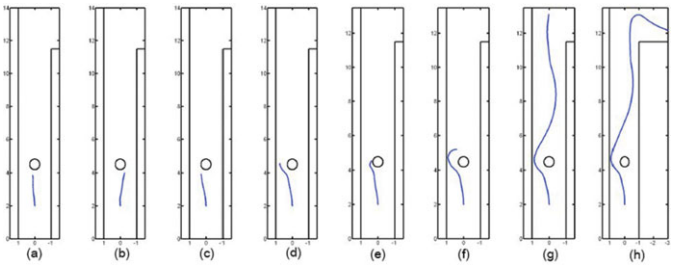

Fig. 19. (Colour online) Experimental results of the skill of passing a hallway when there is a circle-like obstacle. (a) Trial 1. (b) Trial 2. (c) Trial 3. (d) Trial 4. (e) Trial 5. (f) Trial 12. (g) Trial 13. (h) Trial 34.

were represented, the SOS algorithm was initiated with $\alpha = 0.5$ and $\gamma = 0.5$, and at each state, if no failure happened, $r = 0$; otherwise $r = -100$. Simulation results (see Fig. 18) show that the robot could not pass the hallway in the beginning when there was a circle-like obstacle, and after 10 trials on the average, the new skill was self-organized and the robot was able to pass the hallway by utilizing the learned skills of hallway-passing and traveling around a circle. However, the robot bumped against the wall at the end of the hallway after 10 trials because it was in a singular situation of the hallway-passing skill when the robot orientation was perpendicular to the wall. Thus, the SOS algorithm entered into steps **S6**–**S10** to create new SRUs for accomplishing the task (this situation has not been ever learned in experiments 1 and 2). In step **S6**, N_i was 4 for each of the two action responses – robot motor speed and turn rate. Thus, there were 16 actions of the new SRU candidates in A_{new}. In Fig. 18, the new SRUs were created and learned to make a right turn to avoid bumping against the wall between trials 10 and 26 by the SOS algorithm. After trial 26, the robot entered the possible initial state of the hallway-passing skill. Thus, step **S3** of the SOS algorithm made the robot continue to pass the hallway by utilizing the skill of passing a hallway. The experimental results are shown in Fig. 19. The robot took 35 trials on the average and 3.4 trials on the variance to pass the hallway.

In our experiments, we demonstrated a new skill (passing a hallway with a circle-like obstacle) that was self-organized from the previous learned skills (hallway-passing and circle-traveling skills), and each learned skill was represented by

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

perceptual stimuli (distance to the wall and obstacle) and action responses (motor speed and turning angle). However, in the previous work, they did not perform such a robot task that was self-organized from learned skills. Also, the SRUs were all self-categorized by neuro-fuzzy networks in our experiments instead of manually designed in the previous work. We utilized SRUs to compare relationship between two similar skills (traveling around a circle and traveling around an ellipse) and determine the common SRUs among them. Compared with the previous work, our experiments demonstrated how to keep the pool of SRUs to a minimum by investigating similarity between two skills. Last, the SRUs in our experiments show the advantage over the manually designed primitives in the previous work because the action patterns of the former ones were adjustable through reinforcement learning for accomplishing the tasks but the later ones were fixed during the execution of tasks.

In our experiments, since the computational complexity was mainly caused by Q-learning, it exponentially increased with the number of states of Q-learning. The states were described by SRUs with respect to the perceptual stimuli and action responses. Thus, more SRUs resulted in more computational complexity. To ease the computational complexity, resolution control reduced the initial 124 SRUs to only 8 SRUs in Experiment 1. In Experiment 2, skill synthesis took 16 SRUs instead of 24 SRUs for traveling around a circle and an ellipse. Last, we utilized the current SRUs and few new SRUs to accomplish the new task in Experiment 3.

6. Conclusions

In this paper, we have presented a neuro-fuzzy-based, self-organizing skill-learning framework, consisting of skill decomposition and skill synthesis, to acquire and decompose a skill into a small number of significant SRUs for accomplishing a given task, and self-organize learned skills into a new skill for a new task.

The salient feature of the proposed neuro-fuzzy-based, self-organizing skill-learning framework is its capability of learning a skill by self-categorizing it into significant SRUs and learning a new skill for a new task from learned skills composed of SRUs without human intervention. In this framework, SRUs, realized by fuzzy rules in a neuro-fuzzy network, serve as the fundamental units in skill decomposition and skill synthesis. With the capability of SRUs in abstracting the mapping between perceptual stimuli and action responses, SRUs make it possible to learn new skills from previously learned skills.

The potential limitation of the proposed framework is that a skill represented by SRUs may not have satisfactory performance on the task involving state-dependent executions. Since the structure of SRU is feedforward, using SRUs to represent a skill does not sufficiently exhibit temporal behaviors. Thus, a potential solution to enhance the current framework is to use a feedback structure for SRUs such as recurrent networks. Because of the directed cycle of a recurrent network, a skill represented by a recurrent network has the internal memory to process state-dependent execution.

Extensive computer simulations and experiments on a Pioneer P3-DX mobile robot have validated the self-organizing capability of the proposed skill-learning framework. In the first experiment of a hallway-passing skill, the robot avoided learning the zig-zag actions from the demonstration data. In the second experiment, the results showed how a skill of traveling around an ellipse was transferred to a skill of traveling around a circle by sharing their common SRUs. In the last experiment, a new skill was self-organized from the hallway-passing and circle-traveling skills to pass a hallway with the presence of a circle-like obstacle. Future work involves the extension of the proposed framework to humanoid robots.

Acknowledgments

This work was supported in part by the National Science Foundation under Grants IIS-0427260 and IIS-0916807. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. D. N. Nenchev and A. Nishio, "Ankle and hip strategies for balance recovery of a biped subjected to an impact," *Robotica* **26**, 643–653 (2008).
2. T. Kanda, H. Ishiguro, M. Imai, T. Ono and K. Mase, "A Constructive Approach for Developing Interactive Humanoid Robots," **In**: *Proceedings of the IEEE/RSJ International Conference on Intelligent and Robotic Systems*, Lausanne, Switzerland (2002) pp. 1265–1270.
3. P. Tomporowski, *The Psychology of Skill: A Life-Span Approach* (Praeger, Westport, CT, 2003).
4. H. Mosemann and F. Wahl, "Automatic decomposition of planned assembly sequences into skill primitives," *IEEE Trans. Robot. Autom.* **17**, 709–718 (2001).
5. T. Shibata, T. Abe, K. Tanie and M. Nose, "Motion Planning of a Redundant Manipulator Based on Criteria of Skilled Operators," **In**: *Proceedings of the IEEE International Conference Systems, Man and Cybernetics*, Vancouver, BC, Canada (1995) pp. 3730–3735.
6. G. M. Bonea and M. A. Elbestawi, "Robotic force control for deburring using an active end effector," *Robotica* **7**, 303–308 (1989).
7. J. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Trans. ASME J. Dynamic Syst. Meas. Contr.* **63**, 220–227 (1975).
8. T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, **78**(9):1481–1497 (1990).
9. M. D. Buhmann, *Radial Basis Functions: Theory and Implementations* (Cambridge University Press, Cambridge, UK, 2003).
10. C. Baroglio, G. Attilio, M. Kaiser, M. Nuttin and R. Piola, "Learning controllers for industrial robots," *Mach. Learn.* **23**, 221–249 (1996).
11. M. Nechyba and Y. Xu, "Human Skill Transfer: Neural Networks as Learners and Teachers," **In**: *Proceedings of the IEEE/RSJ International Conference Intelligent Robots and Systems*, Vancouver, BC, Canada (1995) pp. 314–319.
12. Z. Wasik and A. Safiotti, "A Fuzzy Behavior-Based Control System for Manipulation," **In**: *Proceedings of the IEEE/RSJ International Conference Intelligent and Robotics Systems*, Lausanne,Switzerland(2002) pp. 1596–1601.
13. J. Yang, Y. Xu and C. S. Chen, "Human action learning via hidden Markov model," *IEEE Trans. Syst. Man Cybern. A* **27**, 34–44 (1997).

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press

14. G. Hovland, P. Sikka and B. McCarragher, "Skill Acquisition from Human Demonstration Using a Hidden Markov Model," **In**: *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MN (1996) pp. 2706–2711.
15. T. Speeter, "Primitive-Based Control of the Utah/MIT Dextrous Hand," **In**: *Proceedings of the IEEE International Conference on Robotics Automation*, Sacramento, CA (1991) pp. 866–877.
16. G. Milighetti, H. B. Kuntze, C. W. Frey, B. Diestel-Feddersen and J. Balzer, "On a Primitive Skill-Based Supervisory Robot Control Architecture," **In**: *Proceedings of International Conference on Advanced Robotics*, Seattle, WA (2005) pp. 141–147.
17. M. Matarić, "Behavior-based control: Examples from navigation, learning, and group behavior," *J. Exp. Theor. Artif. Intell.* **9**, 323–336 (1997).
18. R. Arkin, "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior," **In**: *Proceedings of the IEEE International Conference on Robotics and Automation*, Raleigh, NC (1987) pp. 264–271.
19. M. Arbib, "Perceptual structures and distributed motor control," **In:** *Handbook of Physiology – The Nervous System II: Motor Control* (V. B. Brooks, ed.) (American Physiological Society, Bethesda MD, 1981)pp. 1449–1480.
20. R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.* **2**, 14–23 (1986).
21. A. H. Purnamadjaja and R. A. Russell, "Pheromone communication in a robot swarm: Necrophoric bee behaviour and its replication," *Robotica* **23**, 731–742 (2005).
22. M. Matarić, V. Zordan and Z. Mason, "Movement Control Methods for Complex, Dynamically Simulated Agents: Adonis Dances the Macarena," **In**: *Proceedings of the 2nd International Conference Autonomous Agents*, Stuttgart, Germany (1998) pp. 317–324.
23. J. Connell and P. Viola, "Cooperative Control of a Semi-Autonomous Mobile Robot," **In**: *Proceedings of the IEEE International Conference on Robotics and Automation*, Cincinnati, OH (1990) pp. 1118–1121.
24. A. Martineza, E. Tunstela and M. Jamshidi, "Fuzzy logic based collision avoidance for a mobile robot," *Robotica* **12**, 521–527 (1994).
25. M. Slack, *Situationally Driven Local Navigation for Mobile Robots*, JPL Publication 90-17 (California Institute of Technology, Jet Propulsion Lab., Pasadena, CA, 1990).
26. C. Connolly, "Applications of Harmonic Functions to Robotics," **In**: *Proceedings of the IEEE International Symposium on Intelligent Control*, Vancouver, British Columbia (1992) pp. 498–502.
27. S. Singh, "Transfer of learning by composing solutions of elemental sequential tasks," *Mach. Learn.* **8**, 323–339 (1992).
28. S. Singh, "Reinforcement Learning with a Hierarchy of Abstract Models," **In**: *Proceedings of the 10th Nat. Conference Artificial Intelligence* (AAAI Press, San Jose, CA, 1992) pp. 202–207.
29. R. Sutton, D. Precup and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.* **112**, 181–211 (1999).
30. R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," **In**: *Advances in Neural Information Processing Systems 10* (MIT Press, Cambridge, MA, 1998).
31. T. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Research* **13**, 227–303 (2000).
32. P. M. Fitts and M. I. Posner, *Human Performance* (Brooks/Cole, Belmont, CA, 1967).
33. R. Proctor and A. Dutta, *Skill Acquisition and Human Performance* (Sage, London, 1995).
34. S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.* **3**, 233–242 (1999).
35. G. S. Dordevic, M. Rasic, D. Kostic and V. Potkonjak, "Representation of robot motion control skill," *IEEE Trans. Syst. Man Cybern. C* **30**, 219–238 (2000).
36. M. N. Nicolescu and M. J. Matarić, "Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice," **In**: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, USA (2003) pp. 241–248.
37. C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.* **40**, 1320–1336 (1991).
38. C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems* (Prentice-Hall, Upper Saddle River, NJ, 1996).
39. J. S. Wang and C. S. G. Lee, "Self-adaptive neuro-fuzzy inference systems for classification applications," *IEEE Trans. Fuzzy Syst.* **10**, 790–802 (2002).
40. M. Setnes, R. Babuška, U. Kaymak and H. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification," *IEEE Trans. Syst. Man Cybern. B* **28**, 376–386 (1998).
41. A. Barto, R. Sutton and C. Anderson, "Neuron-like adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst. Man Cybern.* **13**, 834–846 (1983).
42. C. S. G. Lee and C. T. Lin, "Supervised and Unsupervised Learning with Fuzzy Similarity for Neural-Network-Based Fuzzy Logic Control Systems," **In**: *Proceedings of the IEEE International Conference Systems, Man and Cybernetics* (1992) pp. 688–693.
43. Y. Jin, W. Von Seelen and B. Sendhoff, "On generating FC^3 fuzzy rule systems from data using evolution strategies," *IEEE Trans. Syst. Man Cybern. B* **29**, 829–845 (1999).
44. A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes* (McGraw-Hill, Columbus, OH, 2001).
45. D. Janglova, "Neural networks in mobile robot motion," *Int. J. Adv. Robot. Syst.* **1**, 15–23 (2004).
46. Z. Anmin and S. X. Yang, "Neurofuzzy-based approach to mobile robot navigation in unknown environments," *IEEE Trans. Syst. Man Cybern. C*, **37**610–621 (2007).

https://doi.org/10.1017/S026357471100124X Published online by Cambridge University Press