Using Knowledge Ontologies and Neural Networks to Control Service-Oriented Robots

Wei-Po Lee Tsung-Hsien Yang

***Abstract*—In recent years, researchers have been building home robots able to interact and work with people. However, due to the complexity and diversity of the environments for developing robots, it is difficult to share and reuse robot code created by different providers. In this work we present a service- based approach that exploits the standard web interface to develop reusable robotic services. Our approach includes knowledge ontology and neural network learning strategies for robot control. In addition, several service functions, including service discovery, selection, and composition have been developed for operating the services. The proposed approach has been implemented and evaluated, and the results show that it can be used to build robotic services successfully.**

I. Introduction

Developing home service robots to achieve user-specified tasks has become a significant issue for people's future life. However, due to the complexity and diversity of the environments for developing robot, it is often difficult to integrate and share the robot application software (such as services) constructed by different providers. This has therefore impeded the development of service robots [1].

To overcome the above difficulties, researchers have been building robot design frameworks to manage the complexity and facilitate the reusability of robot code. From the point of view of the end-users, they expect to obtain and use robot application software conveniently, rather than to care about the design details. Then, from the point of view of the robot designers, they prefer an easy-to-share environment in order to integrate application services constructed by different providers. Taking into account the needs of both sides, service-oriented architecture (SOA, [2]) provides a promising option for developing robotic services. One of the major advantages of SOA is that the shared resources are available on demand. It means that these resources can be regarded as independent services and accessed without knowledge of their underlying platform implementation [1][2]. Ideally, with a SOA-based robotic framework, end-users can control their robot in the similar way of using web services, and robot designers can share services with each other and reuse available code to design more comprehensive services.

However, unlike the traditional web services, applying SOA to robot applications involves the complicated control of robot actions, and thus the design of robotic services becomes challenging. To develop robot code to solve application tasks, a common way is to adopt a divide-and-conquer strategy. The process of dividing and solving robotic task is similar to that of task decomposition and service composition in the web service domain, in which most problems can be solved by workflow-based or AI planning methods [3][4]. Hierarchical Task Network (HTN) planners are typical examples [5]. Such kind of "hand- tailorable" planners indicate the current trend of combining automatic planners with human efforts to generate composite services. With the aid of an ontology defined to describe robot tasks and to guide the task decomposition, the methods used in web composition can also be applied to developing of composite robotic services to solve complicated tasks.

In this work, we present a robotic service framework that aims to fulfill the needs of end-users and robot designers. Our work employs a service-oriented architecture in which a robot controller is created and regarded as a service, and the complicated robot tasks are achieved through the composition of available services. It includes a neural network-based learning mechanism to create new robotic services, and integrates the web services description language OWL-S (Ontology Web Language-Services [6]) and HTN planners to perform ontology-based service composition. To verify the system and demonstrate how it works, we have conducted different sets of experiments in which the robot can successfully achieve user-specified control tasks in a home environment.

II. Related Works

In the study of service-oriented computing and modeling, web service technologies have been widely applied to different types of applications. With many successful experiences, researchers are now pressing on to extend these techniques to the development of robot systems. The most related works are the ones regarding robots as services and exploiting the web service architecture to create robots. For example, Yachir *et al.* employed service composition techniques to plan robotic services to help an elderly person [7]. Kim *et al.* focused on how to control a robot through the integration of web service and robot application technologies [8]. Similarly, Ha *et al.* proposed a service-oriented architecture for the integration of ubiquitous robot devices, including sensors and motors [9]. There are also other works integrating robotic and web technologies. For example, in a recent research conducted by Blake *et al.*, robots can be equipped with web-oriented software interfaces that help them to access universally standard web resources [10].

Different from these studies, our work dedicates to construct the explicit task ontology for daily home tasks, and focuses on the planning-based service configuration to obtain robotic services. In particular, we develop a practical method to create new services and this issue has not been addressed in the work described above. Our method is a kind of demonstration-based programming, which is to teach a robot

Wei-Po Lee and Tsung-Hsien Yang are both with National Sun Yat-sen University, Kaohsiung, Taiwan (contact e-mail: wplee@ mail.nsysu.edu.tw).

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

how to achieve a task through human demonstration. Our work represents a task as the behavior trajectories at the data level, and it includes a neural network learning procedure to derive controllers.

III. A Service-Oriented Approach for Robot Control

A. System Framework

Service-oriented computing (SOC) has become a trend of contemporary programming. It regards services as self-described reusable building blocks that can be used to support the development of software applications. Following the SOC design principles, we present a service-oriented framework that is implemented by the standard-based, platform-independent techniques (such as SOAP, REST). This framework can exploit the corresponding advantages of SOC to provide rapidly prototyping robotic services.

Fig. 1 illustrates the conceptual framework with operating flow of the proposed robotic service system. As can be seen, our system includes three major modules and a task ontology working tightly with these modules. The first module is a machine learning mechanism, with which a user can conveniently derive robot control code to create his services. The second module is responsible for service discovery and selection. It analyzes the command (or query) specified by the user and searches the service repository accordingly. The task ontology shown in the figure is built for command interpretation and service mapping. It describes the semantics of a task in terms of task structure and task-solving process. Then this module employs a pre-defined selection strategy to choose the most suitable service from the candidates. The third module is developed for service composition. It decomposes the target task into several subtasks in a recursive manner according to the specification of the task ontology, and finds relevant services for each subtask. This module then integrates these services through a composition procedure to achieve the target task. The user can observe the robot behavior through a pre-designed interface. Based on the user's evaluations, this module can repeatedly perform configuration procedure to find new services to satisfy the user's needs. The details are described in the subsections below.

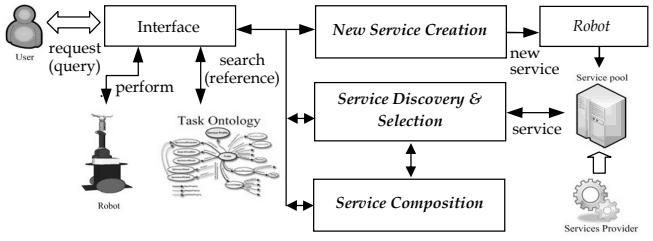

Fig. 1. System overview.

B. Using Knowledge Ontology to Identify Robot Task

Ontology plays an important role in knowledge representation. For an action planning problem (e.g., a robot task), ontology can be defined in order to provide the sequence of problem-solving steps through organizing domain knowledge. The planner (i.e., problem solver) can then exploit domain knowledge and the related techniques defined within the ontology to achieve the application task. In this work, we construct two ontologies, task ontology and position ontology, to specify the structure of the problem-solving process and to describe the environmental knowledge for the robot, respectively. To accomplish the target task, a service robot should understand both ontologies.

The first ontology, position ontology, defines the locations of different objects in a home environment and the containment relationships between the objects. For example, this ontology can indicate that the bowls, plates, and cups are put on a cupboard, and a TV is placed in the living room. And the second ontology, task ontology, describes how to resolve a user's request by a sequence of steps. It interprets the task complexity in a hierarchical way. Fig. 2 shows a part of the task ontology defined in the proposed system. The task ontology has included the possible robot actions/tasks (i.e., what the robot can perform) in the environment so that the robot can follow it accordingly to achieve the target task.

The procedure for using task ontology is that, after receiving a control command, the system will parse the command to extract the verb part as the action description, and then use the description to match the task terms recorded in the task ontology. Currently, we take a simplified natural language toolkit (i.e., NLTK, [11]) for command parsing. For example, the system will analyze a command "give me a cup" to get the verb "give" at first, and then use the word "give" to match the terms included in the task ontology.

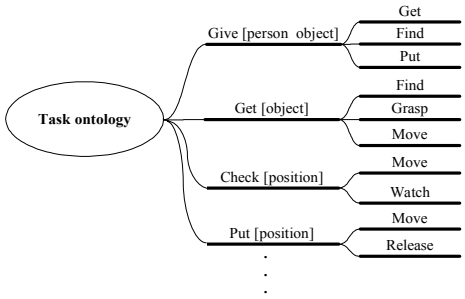

Fig. 2. Part of the task ontology.

To enhance the problem-solving capability of the task ontology, we use the powerful ontology description language OWL-S to implement the ontology. It has three major parts with essential types of knowledge: service profile, service model, and service grounding. In general, *service profile* provides the information to express "what the service does", including a description of what to be accomplished, and the limitations and requirements of the service. S*ervice model* describes "how to use the service", which is similar to the problem solving process for annotating a robotic service. It includes information about inputs, outputs, preconditions and effects, in order to perform analysis, composition, activities, and monitoring. And *service grounding* presents "how to access a service". It is to specify a message format, a communication protocol, and other implementation details. Fig. 3 shows an OWL-S example that describes the service model defined for the robot behavior "Give", which is composed by one atomic process "Find" and two composite processes "Get" and "Put". The relevant details of this application will be given in the experiment section.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
<process:CompositeProcess rdf:ID="Give">
  <process:hasInput rdf:resource="#Person"/>
  <process:hasInput rdf:resource="#Object"/>
  <process:hasPreconditoion rdf:resource="#ExistPerson"/>
  <process:hasPreconditoion rdf:resource="#ExistObject"/>
  <process:hasEffect rdf:resource="#PersonHasObject"/>
  <process:composedOF>
    <process:Sequence>
      <process:components rdf:parseType="Collection">
        <process:CompositeProcess rdf:about="#Get"/>
        <process:AtomicProcess rdf:about="#Find"/>
        <process:CompositeProcess rdf:about="#Put"/>
      </process:components>
    </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>
```

Fig. 3. The OWL-S example of a robotic service model.

C. Creating a Monolithic Robot Service

The core of creating a new service is to develop the service function, which can then be combined with other relevant service descriptions and specifications to constitute a service. The service function means the control code for driving the robot to act in the environment to achieve a task. The control code can be written manually by a programmer or alternatively be learnt without explicit programming. The latter is a more intuitive and natural way for ordinary users and it can thus save their efforts in programming a robot.

To construct robot controllers automatically, we develop an approach of programming-by-demonstration, in which perception and motion information of the behavior sequences demonstrated by the user are first recorded, and then a machine learning mechanism is used to derive controllers. Fig. 4 illustrates the procedure. The robot is driven manually to achieve the target task. During the period of human-driven demonstration, at each time step the relevant information is recorded to form a behavior data set for later training. In other words, it is to derive the time-series profiles of perception and motion information from the qualitative behavior shown by the robot. After the behavior data is obtained, in the second stage the robot plays the role of a learner that is trained to achieve the target task.

In this work, a recurrent neural network (RNN) model is adopted as a behavior controller for the learner. Here, we take a fully connected RNN architecture as the robot controller, and implement a learning mechanism to train the controller. When the model is used to control a robot, each network node corresponds to an actuator of the robot in principle. Also, two extra nodes are added to serve as buffers, and their roles are not specified in advance. The redundancy makes the controllers easier to be learnt from data.

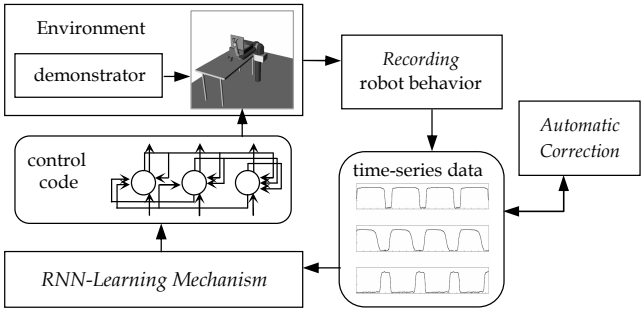

Fig. 4. The procedure of programming-by-demonstration.

To find the settings of a neural network (i.e., thresholds, time constants, weights), we adopt the back-propagation through time learning algorithm (BPTT, [12]) to update the relevant parameters of recurrent networks in discrete time steps. The goal here is to minimize the accumulated discrepancy between the time-series data recorded in the demonstration procedure (i.e., desired values) and the values produced by the control model (i.e., actual values). The above error function can be defined directly as the mean squared error over the time period:

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\left\{\frac{x_i^a(t)-x_i^d(t)}{x_i^d(t)}\right\}^2$$

In the above equation, $x_i^d(t)$ is the desired output of node i at time t, $x_i^a(t)$ is the value generated from node i of the learnt model, N is the number of nodes in the network, and T is the number of time points for data collection. Also, the heuristic method delta-bar-delta is used to modifying the learning rate in the training procedure.

Though the above learning procedure is an automatic and efficient method to obtain robot controllers, to ensure the success of this approach, the problem of class imbalance has to be encountered [13]. This is a crucial problem in machine learning community as the data collected is often distributed unequally in the real world applications. Corresponding to a classification task, here the pair of input (perception information) and output (motion information) of the controller at a certain time step is used to represent a data point, and each performing-skill for achieving a goal behavior is regarded as a specific class that considers similar data points (i.e., training instructions). In a demonstration- based method, the data points collected in the demonstration procedure are often distributed unequally in different classes, and the target controller thus cannot be learnt perfectly.

One popular way to resolve this problem is to collect new data through having the demonstrator iteratively present the parts the robot is yet to learn. The newly obtained behavior sequences are then added to the training set to change the corresponding data distribution. After that, the re-learning procedure is started again to build a new controller. To reduce the user's workload in iterative demonstrations, we take the viewpoint of data analysis to develop a mechanism that can automatically correct behavior data in the test trials to create a new training data set. In our approach, the data points in the test trials are categorized into two types, based on whether a critical error occurs at the points. Different correction rules are defined and applied to the two types of data points accordingly. We will take an example in the experiment section to illustrate how the proposed approach works.

D. Service Operations

In this work, four types of functions, including service discovery, selection, composition, and reconfiguration, are developed to enable the available robotic services to achieve user-specified command. The first type of functions is service discovery. It is the process of searching available services to find the ones that can fulfill the user's requests. As mentioned, using OWL-S to describe services can standardize the service description in semantics. In this way, an automatic method can be developed to find appropriate services, according to the semantic descriptions associated with the services. In our system, the OWL-S service profile defines the IOPE elements

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

(i.e., Input, Output, Precondition, and Effect), which are used to match the user's service requests afterward. The procedure is that if the control command can match the task terms defined in the ontology, the system will collect the relevant information (i.e., IOPE) corresponding to this task term to examine whether the information matches that of the services advertised in the service repository. A successful match means that the system can find services to fulfill the user's request. For instance, in the task ontology shown in Fig. 2, the task "Get" has the input "target object name", the precondition "target object exists", and the effect "target object is on hand". Once the task "Get" matches the user command, the task descriptions are used to find the corresponding services.

The second type of functions, service selection, is choosing the most suitable services for the target task from the candidates provided by the service discovery process. To select the best service, many researchers have defined various attributes (such as accessibility, cost, response time, reputation, etc. [14]) and proposed different quality-of service (QoS) based selection methods. Among these attributes, service reputation represents the direct evaluation results from the service requesters or the neutral party. It is an objective and easy-to-measure attribute. Therefore, in our current implementation, we choose to use this attribute for service selection, and design a rating strategy to measure service reputation. It is to combine all ratings (from the lowest 1 to the highest 5) given by user to rank the candidate services obtained from the service discovery process. The system will then select services according to the ranking order, in response to the user's request.

The third type of functions is service composition, which is to automatically compose services already existing in the repository to achieve more complex tasks. In the proposed framework, we adopt the AI planning techniques for service composition. Among others, HTN planning is a well-designed methodology most suitable for the service composition. In the HTN planning domain, the tasks can be categorized into two types: the primitive and the compound. A primitive task can be performed directly by the predefined planning *operators*, while a compound task needs to be decomposed by a planning *method* before being performed. The latter performs task decomposition according to the task ontology described in section III.B to break the original task hierarchically, and then the planner solves the subtasks in the reverse order and produces a sequence of actions for the original task.

The concept of task decomposition in HTN is in fact very similar to the composite process decomposition in the OWL-S service model: it reduces the complexity of reasoning by eliminating uncertainty during the planning process. One well-implemented HTN planner is the Simple Hierarchical Ordered Planner 2 (SHOP2, [15]). The authors have proved the semantic correspondences between the SHOP2 planning and the situation calculus of the OWL-S process model [5]. This means that the HTN planner can be an efficient tool to work with the hierarchically structured OWL-S process model. Following their studies, here we choose to employ SHOP2 to conduct our robotic service composition and take the task ontology defined above to guide the decomposition.

As OWL-S and SHOP2 have their own internal representations, to use SHOP2 for service composition, translations between OWL-S and SHOP2 need to be defined. Sirin et al. have defined some translations, from the OWL-S process models to the SHOP2 domains and from the OWL-S composition tasks to the SHOP2 planning problems [5]. With such transitions, a service originally created by the OWL-S descriptions can have its SHOP2 format and be used by the planner for service composition.

To realize the service reconfiguration, we use the software OpenRAVE ([16]) to construct simulated robots and the environments the robots are situated in. In our present work, when the system executes a composite service in OpenRAVE, the planned action steps are also presented in the robotic service interface. So the user can compare the planned steps with the robot actions to examine whether the robot has achieved the task successfully. If the user finds that the actions do not match his request in simulation, he can ask the system to conduct the service composition again. The user can also create new services to fix an incomplete plan through the learning mechanism described previously.

IV. Experiments And Results

To evaluate the proposed methodology for robot control, two series of experiments have been conducted. The first series is to investigate whether the controllers can be learnt from our demonstration procedure. The second series is to examine how our service composition module can be used to achieve a more complicated task at a higher level.

A. Creating New Services through Robot Learning

In the first series of experiments, a robot arm with a camera mounted on the arm has been used to achieve a sequential task of opening a box and picking-up a cup inside the box. To save the evaluation time, the experiments were performed in simulation. In our experiments, a fine time-slice technique was used in simulation and each time step lasted for 100 ms. Initially, a box was put on the table. The robot needed to open the box and decided what to do next according to what it observed. If the robot found a cup in the box, it was required to pick up the cup, put the cup on the table, and then close the box. If the box was empty, the robot simply closed the box. After the box was closed, the robot had to decide to take which of the following two actions: whether to open the box (in order to check the box and pick up the cup if there was any) or to move back to its initial position (the task had been completed). As can be observed, the robot was not able to make the correct decision relying only on the perception information, but not the previous task state alone. It must integrate both types (i.e., the perception and the internal state) of information in order to achieve the task correctly.

The control mechanism for the above sequential task can in fact be considered as a finite state machine that includes four internal states to represent respectively the task status below: (1) the box is on the table and the robot is in its initial position; (2) the box is on the table and robot has moved to the position ready for operation; (3) the box has been opened by the robot; and (4) the box has been closed by the robot. For this task, the perception input to the finite state machine is one of the three situations (the box is close, the box is open and the cup is in the box, and the box is open and it is empty) derived from the visual results that have been processed by the camera on the robot arm. Therefore, the control task is to deal with the state transitions based on both the current state and sensor information, and then to generate an appropriate action. Here, the possible actions for the robot are to move to the position

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

ready for operation, to open the box, to pick up the box, to close the box, and to move to the initial position.

To obtain a controller that can produce the expected sequential behavior, the proposed approach has been used to learn a six nodes network. Fig. 5 illustrates the behavior generated by the controller. As we can see, a sequential controller can again be learnt to achieve the task successfully.

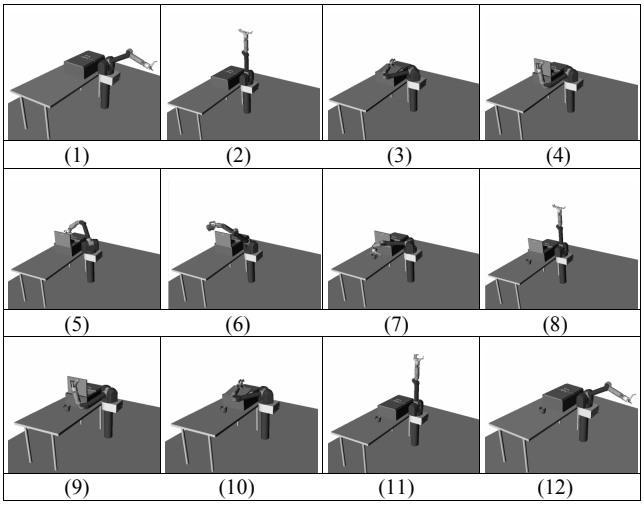

Fig. 5. The behavior sequence produced by the controller successfully learnt.

B. Achieving Service Composition

In the second series of experiments, we use the service system to derive a composite service to achieve an application task in simulation, in which the user asked the robot to give him a cup in a domestic environment. According to our design, the user needs to select the robot type and the task environment from the pre-defined interface, before sending a control command to request robotic services. The robot type is used to derive some hard constraints (e.g., hardware restrictions) that need to be satisfied in the service discovery process; while the task environment provides the initial world states (that are the preconditions of the desired service) and indicates the initial positions of the objects in the environment. In addition to the pre-defined choices, users are allowed to take the XML descriptions to create their own simulated robots and environments.

To understand the user's request, the robot needs to connect to the ontology server for parsing the command "give me a cup". In this application, the system used a natural-language parser to check the syntax of the command and generate a syntax tree. It then sent the verb part to the task ontology and the noun part to the position ontology to search for suitable services. As in the part shown Fig. 2, we can see that to achieve the "Give" task, the system needs two input parameters, namely "Person" and "Object", and it takes the segments "me" and "cup" extracted from the command to correspond to the two parameters, respectively.

The OWL-S example shown in Fig. 3 describes the service model defined for the robot behavior "Give", which is composed by one atomic process "Find" and two composite processes "Get" and "Put". That is, the task "Give an object to a person" can be decomposed into three subtasks: "Get the object", "Find the person", and "Put the object to the person (meaning location here)". In addition, the process for "Give" has two inputs "Person" and "Object" (to tell the robot which object to take, and to whom) and two preconditions "ExistPerson" and "ExistObject" (to indicate if any person or object exists in the world state).

With the decomposition result, the system can translate the "Give" service described by OWL-S into a HTN *method*, and invoke the sub-services included in the "Give" process to complete the overall task. Fig. 6 shows the HTN *method* corresponding to the "Give" service, which includes the subtasks "Get", "Find", and "Put". The action steps of "Give me a cup" can be carried out by the relevant atomic services (i.e., HTN *operators*), in the order of the following verb-object pairs: (find cup), (move table), (grasp cup), (find user), (move user), and (release cup).

Fig. 7 illustrates how the robot achieved the task in the two test scenarios in simulation. The first four steps in the figure (i.e., steps (1)-(4)) show that the robot used the position ontology to infer where the cup was located. For the first scenario, once the robot knew that the cup was in the kitchen, it went there and recognized that the cup was put on the table. Then the robot moved to the region around the cup so that it could grasp the cup. Steps (5)-(7) describe such a situation: the robot moved to a position close to the cup, and performed the "Get" service. Next, steps (8)-(12) show that after the robot picked up the cup, it invoked the other two services "Find" and "Put" to find the user and give him the cup.

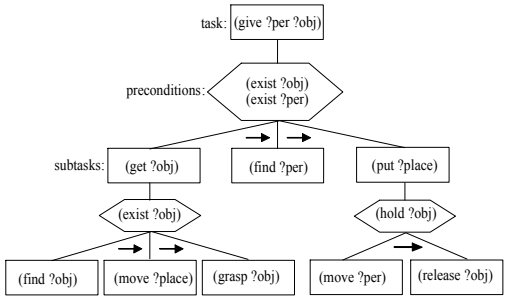

Fig. 6. The HTN method for the target service.

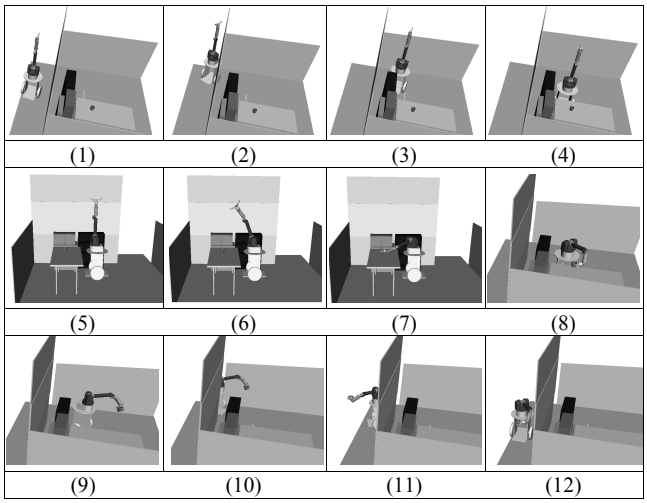

Fig. 7. The simulation results for the application task.

V. CONCLUSION

In this work, we presented a service-oriented approach that includes knowledge ontology and neural network learning

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

strategies for robot control. The task ontology has been constructed and used for command interpretation and service mapping, and the neural network has been used to create new services through a programming-by-demonstration procedure. In addition, several service functions, including service discovery, selection, and composition have been developed for operating the services. Under the guidance of task ontology, our work employs a planning-based service composition process to generate composite robotic services to solve complicated tasks in the home environment. Experiments have been conducted to verify the proposed methodology, and the results show that new services can be built through human-machine interaction and composite services can be derived for the application task successfully.

REFERENCES

[1] Amoretti, M., Reggiani, M., 2010. Architectural paradigms for robotics applications. *Advanced Engineering Informatics*, 24(1), 4-13.

[2] Erl, T., 2005. *Service-Oriented Architecture* (*SOA*): *Concepts, Technology, and Design*. Prentice Hall.

[3] Rao, J., Su, X., 2005. A survey of automated web service composition methods. In: *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition,* pp.43-54.

[4] Sun, S.X., Zhao, J., 2012. A decomposition-based approach for service composition with global QoS guarantees. *Information Sciences*, 199, 138-153.

[5] Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D., 2004. HTN planning for web service composition using SHOP2. *Web Semantics: Science, Services and Agents on the World Wide Web,* 1(4), 377-396.

[6] Martin, D., Hobbs, J., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K., 2004. OWL-S: semantic markup for web services. *W3C Member Submission,* 22, 2007-04.

[7] Yachir, A., Tari, K., Chibani, A., Amirat, Y., 2008. Towards an automatic approach for ubiquitous robotic services composition. In : *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.3717-3724.

[8] Kim, B.K., Miyazaki. M., Ohba, K., Hirai, S., Tanie, K., 2005. Web services based robot control platform for ubiquitous functions. In: *Proceedings of IEEE International Conference on Robotic and Automation*, pp. 691-696.

[9] Ha, Y.G., Sohn, J.C., Cho, Y.J., Yoon, H., 2007. A robotic service framework supporting automated integration of ubiquitous sensors and devices. *Information Sciences*, 177(3), 657-679.

[10] Blake, M.B., Remy, S.L., Wei, Y., Howard, A.M., 2011. Robots on the web. *IEEE Robotics and Automation Magazine,*18(2), 33-43.

[11] Loper, E., Bird, S., 2002. NLTK: the natural language toolkit. In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp.63-70.

[12] Werbos, P. J., 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78, 1550-1560.

[13] Liu, X.Y., Wu, J., Zhou, Z.-H., 2009. Exploratory under-sampling for class-imbalance learning. *IEEE Trans. on Systems, Man and Cybernetics*, *Part B*, 39, 539-550.

[14] Yu, T., Zhang, Y., Lin, K.J., 2007. Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Trans. on the Web*, 1(1), article number 6.

[15] Nau, D., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D., Yaman, F., 2003. SHOP2: an HTN planning system. *Journal of Artificial Intelligence Research*, 20, 379-404.

[16] Diankov, R., Kuffner, J., 2008. OpenRAVE: a planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)