Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

KeJia: towards an autonomous service robot with tolerance of unexpected environmental changes*

Wei SHUAI‡, Xiao-ping CHEN

Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China

E-mail: swwsag@mail.ustc.edu.cn; xpchen@ustc.edu.cn

Received Feb. 20, 2019; Revision accepted Mar. 20, 2019; Crosschecked Mar. 27, 2019

Abstract: KeJia is a domestic service robot, consisting of a mobile base, an arm, two cameras, and a set of software components for perception, manipulation, natural language understanding, motion and task planning, and decision making. With on-line running of these functions, a robot can adapt to dynamic environments which may have unexpected changes. In this paper, we propose a novel hierarchical method which combines motion planning with a neural network, so that the robot can tolerate errors from sensors, wear of parts, and human disturbances during motion execution. We evaluate our work on KeJia that cooks popcorn using a microwave oven, where humans try to disturb KeJia during the operation.

Key words: Robot; Task planning; Manipulation
https://doi.org/10.1631/FITEE.1900096 **CLC number:** TP242.6; TP399

1 Introduction

Robotics is a cross-disciplinary technology involving mechanical design, hardware control, sensing, artificial intelligence, etc. In recent years, domestic service robots have received much attention from researchers. A classic task for service robots is receiving commands from users (e.g., grasp the coffee from the dining table for me). Compared with traditional industrial robots, domestic service robots in home scenarios have significant differences:

1. The operating environment for industrial robots is highly structured and customized. In operation, a vast majority of industrial robots prohibit persons or unrelated objects from entering the workspace to avoid accidents. This is different from the environment in home scenarios, where the environment may be affected by persons or pets at any time. Domestic service robots are autonomously required to sense the environmental changes.

2. Industrial robots are generally designed to interact with professional operators through teach pendants, while service robots in home scenarios interact with normal users using touch screens on the robots or natural language in daily life.

3. Industrial robots often ensure high precision through strict customized environmental arrangements, sophisticated instruments, and regular professional maintenance. However, normal users do not have the ability to professionally maintain service robots. Service robots have the ability to autonomously tolerate errors caused by the accumulated use of the process, such as deformation of joint and looseness of screw, using devices with lower precision.

In this study, we present the KeJia service robot designed for home applications and focus on the effort to ensure that the robot is tolerant of unexpected changes. We use an answer set programming (ASP) (Lifschitz, 2008) based integrated decision-

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. U1613216 and 61573333) and the Fundamental Research Funds for the Central Universities, China

ORCID: Wei SHUAI, https://orcid.org/0000-0002-8362-4331

making strategy, which enables KeJia to work in an unstructured and constantly changing environment. KeJia can continuously sense the environment, verify the effectiveness of actions, and replan when the environmental changes affect the execution of the tasks. We also propose a hierarchical method for KeJia's manipulation, combining traditional motion planning methods with a learning-based end-to-end approach using a convolutional neural network (Krizhevsky et al., 2012) which allows the robot to tolerate errors in sensors, link joint clearance, and human disturbances during motion execution.

2 Related works

In the past decade, one of the most popular robots was Honda's humanoid service robot Asimo (Sakagami et al., 2002), which can autonomously walk in an indoor environment, detect people, understand instructions, and carry out tasks (such as pouring coffee). Willow Garage's PR2 robot was a domestic service robot platform on which researchers around the world have carried out significant research, such as folding clothes. KAIST's Hubo robot won the 2015 DARPA Robotics Challenge, which was focused on robot control and manipulation. Works focus on one aspect of the robot's functions, but it is essential to integrate those basic functions into a robotic system.

For integration, the state machine (Bohren and Cousins, 2010) is one of the most classic methods. However, in complex and variable environments, state machine methods are difficult to extend. Using the task planning module is a general approach (Chen et al., 2010; Erdem et al., 2012). Chen et al. (2016) used a continuous observation mechanism that allows the robot to handle error messages.

For manipulation, under the assumption of the ideal geometric model, the robot has mature control systems based on dynamics and kinematics (Vannoy and Xiao, 2008; Murray, 2017). However, an accurate model of the robot, which directly affects the execution result of actions, incurs a high maintenance cost. In a learning-based approach, Pinto and Gupta (2016) and Levine et al. (2018) used neural networks to learn strategies of grasping. However, data collection cost a lot, and generally took months. Mahler et al. (2017) and Popov et al. (2017) trained robots using simulators, but strategies cannot be directly ported to physical robots.

3 Scenario features

Compared with traditional industrial scenarios, home scenarios are unstructured, where the robot and human live in the same environment. Based on our years of experience in domestic service robot research, service robots with the characteristics of adaptation to a dynamic environment and error tolerance perform better in home scenarios than robots that do not have.

3.1 Adaptation to a dynamic environment

Distinct from industrial robots which have isolation zones separated by isolation belts, service robots in home scenarios share the same workspace with operators. While a robot is completing its tasks, people may disturb or assist the robot, making the environment different from the robot's belief. If the robot can adapt to environmental changes in the workspace, it will have better performance in home scenarios.

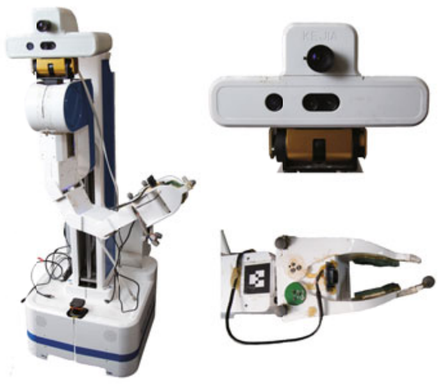
3.2 Error tolerance

Ordinary users do not have the ability to operate the internal program of robot, nor do they have the ability to provide regular maintenance. During daily use, there are unexpected errors of slight looseness and misalignment on hardware, especially on the structure of the robot's arm and the pre-calibrated position of sensors. If the robot cannot deal with those errors by itself, a calibration process operated by professionals is required to re-program parameters of the robot's model. However, if the robot's service quality is not affected by these anomalies, it will perform better in home scenarios than robots that require regular maintenance.

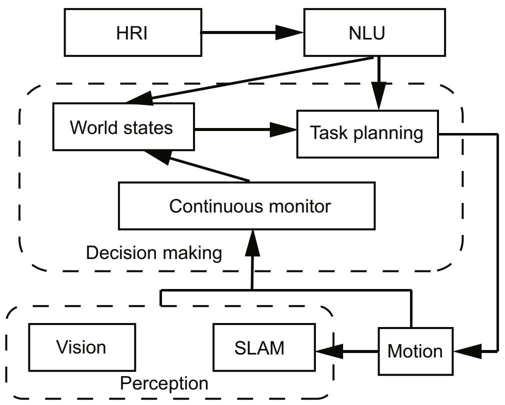
4 Framework overview

4.1 Hardware structure of KeJia

KeJia is a domestic service robot designed for home scenarios, and has shown its accomplishment in the RoboCup@Home Competition (Wisspeintner et al., 2009; Chen et al., 2014). The hardware structure of KeJia is shown in Fig. 1a. A differential wheel drive chassis installed on the middle axis is responsible for movement. KeJia is equipped with a five-degree-of-freedom (5-DOF) arm for manipulation and a two-DOF head for perception in vision. For real-time perception, KeJia is equipped with a Kinect camera which can detect depth information, a high-resolution CCD camera for objects' details, a macro camera attached to a gripper, a large-range HOKUYO UTM-30LX laser scanner which can provide information about walls and furniture for mapping and self-locating, and a microphone.



(a)



(b)

Fig. 1 Hardware and software architectures of KeJia: (a) hardware structure; (b) software architecture

4.2 System architecture of KeJia

KeJia adopts a distributed structure, and all modules in KeJia communicate with each other by the robot operating system (ROS) (Quigley et al., 2009; Chen et al., 2014). In general service scenarios, KeJia takes natural language commands in English as inputs. Once a command is given, the human robot dialogue module will analyze the command and translate it into a proper goal representation in answer set programming (ASP) form (Lifschitz, 2008). As for the task planning module, a solver is used to generate plans with atomic actions and predict states for each step of the actions. Those states are fed into the motion module and the continuous monitor, while the monitor continuously senses world states of the environment using the information from sensors. Every time the robot finishes executing the trajectory obtained from the manipulation module or detects a failure flag from the manipulation module, the continuous monitor will check whether world states and predicted states are consistent. Once a difference is detected, the current plan is aborted and a replanning process is performed. The system architecture of KeJia is shown in Fig. 1b.

4.3 Function modules

4.3.1 Mapping, locating, and navigation

KeJia's navigation module can be divided into three parts: mapping, locating, and navigation. The mapping part maintains information about obstacles in the scene through a prior exploration by professional operators. The locating part informs the robot where it is located. The navigation part is used to control the robot to move to its goal without contacting the obstacles. Both lasers and the depth camera are used in these modules. Particle filter GMapping (Grisetti et al., 2007) is used for mapping and locating. Instead of traditional representation of a two-dimensional (2D) grid map, we use a quadtree-structed map (Chen et al., 2015), which can save a lot of memory space, meaning that the modules can process more particles at the same time when mapping and locating to obtain accurate map information. Two planners are employed for navigation. A global planner uses the A* algorithm to roughly plan way points, and a local planner uses the VFH* (Ulrich and Borenstein, 1998) algorithm to obtain the exact direction along which the robot needs to move.

4.3.2 Dialogue understanding

Dialogue understanding is used for human-robot interaction, including speech recognition (SPR) and natural language understanding (NLU). SPR is responsible for converting the voice information captured by the microphone into strings, and NLU is used to translate strings into semantic representation and output formal representation in the ASP language format. There are three types of commands from users in dialogue understanding: tasks that the robot is about to perform (e.g., grasping the cup), constraints needed to be observed in the robot's movement (e.g., don't enter the bedroom), and supplements to the world state (e.g., drinks are on the dinning table). In the process of semantic parsing, we first use the Stanford parser (Klein and Manning, 2003) to obtain the syntax tree structure, and then use λ-calculus (Blackburn and Bos, 2005) to calculate the semantics of the sentence.

4.3.3 Vision

KeJia's vision module can sense the position and the types of objects which have been pre-trained through template matching with high accuracy in real time. A Kinect depth camera and a high-resolution color camera are invoked as visual sensors, which enable KeJia to obtain real-time depth point clouds for objects' positions and shapes and high-resolution images for texture features of the objects' surfaces. After calibration, a transformation matrix can be established between point clouds and images. While detecting objects, it uses point clouds with the Point Cloud Library (Rusu and Cousins, 2011) to extract the largest horizontal plane and cluster point clouds above it into different pieces, which are possible candidates of objects. Then, KeJia maps the corresponding areas of the high-resolution image from candidates of point clouds and extracts SURF features, HSV color features, size features, etc. After a comparison with the existing features, the vision module can obtain the objects' categories and positions.

5 Integrated decision making

KeJia's decision-making strategy is based on the ASP, representing the transformation about world states which can describe the environmental information and belief states which can describe the robot's beliefs. An ASP solver generates a sequence of actions and states from the initial state to the target state, and drives the robot to complete the specified tasks step by step.

An ASP program consists of a set of ASP rules, expressed as

$$H \leftarrow p_1, \ldots, p_k, \text{not } q_1, \ldots, \text{not } q_m.$$

Different from classical negation "$\neg$" in classical logic, not q means that q cannot be deduced from the ASP program. The ASP rule is equal to the following logical expression:

$$H \vee \neg p_1, \ldots, \vee\neg p_k, \vee\neg\text{not } q_1, \ldots, \vee\neg\text{not} q_m.$$

It is an ASP fact if the ASP rule has only H in its expression. An action is represented by occurs(a, t), where a is the action's name, and t is the time when the action occurs. The ASP fact occurs(a, t) is true only if action a occurs at time step t. A state is represented as $h(s(x), t)$, where x means an object, s the state of the object, and t the time when the state occurs. The ASP fact $h(s(x), t)$ is true only if entity x has state s at time step t. For tasks and constraints from the dialogue understanding module, semantics will be translated into non-monotonous ASP representations (e.g., $\leftarrow$ not goal$(A) \leftarrow$ constranint(A)). For information about the states of the environment, the semantics are converted to the world state, such as obj(A).

The depiction of the knowledge can be divided into three parts: action's conditions, action's influences, and inertia rules. State inertia indicates what happens when the time elapses from t to $t+1$. The action preposition describes the premise that an action is allowed to occur. The action effect represents the transformation of world states when an action successfully runs without any fault.

For example, as shown in Fig. 2, knowledge about action open can be described in the following rules:

1. Influences: If A is an oven and the action open occurs in step t, A will has the state door_open. It means the oven's door will be opened if the action open occurs.

2. Conditions: If A is an oven and the action open occurs in step t, A should have a state door_close in step $t-1$. It means the oven's door must be closed before the action open occurs.

3. Conditions: If A is an oven, B is an object, and the action open occurs in step t, B should have a state outagent in step $t-1$. It means the robot's gripper must be empty before the action open occurs.

4. Inertia rules: If A is an oven, the action open does not occur in step t, and A has a state door_close in step $t-1$, A will have a state door_close in step t. It means the door will stay closed if the action open does not occur.

```
h(door_open(A), t) ← occurs(open(A), t), oven(A)
← occurs(open(A), t), oven(A), not h(door_close(A), t−1).
← occurs(open(A), t), oven(A), not h(outagent(B), t−1), object(B).
h(door_close(A), t)←h(door_close(A), t−1), not occurs(open(A), t), oven(A).
```

Fig. 2 Examples of ASP rules in the integrated decision-making module

ASP solver's work is to find the set of ASP facts, which lets all of the ASP rules in the ASP program be true. According to the solver and ASP rules, a sequence $< s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n >$ will be given, where s_i is the set of belief states, and a_i is the action the robot plans to produce. Robots may fail in executing such a sequence for two reasons: (1) s_i does not match the actual world states (conditions are not satisfied); (2) action a_i fails during its execution (s_{i+1} cannot be reached). During the execution of the sequence of actions and states, KeJia constantly updates its world states through the perception module, while the manipulation module is continuously monitored for abnormalities, such as excessive joints' current, which indicates that a collision may occur during movement. When belief states that conflict with world states sensing from perception or abnormalities are detected, the current sequence is considered illegal and will be replanned (Algorithm 1).

6 Manipulation

KeJia's manipulation module is studied only for grasping, placing, and touching operations. For these operations, KeJia's behavior can be split into two parts: moving the end effector of the robot's arm to a certain pose, and closing or opening its paw at the previous pose. Consequently, the grasping, placing, and touching operations are equivalent to driving the end effector to fall within a certain range and not colliding with other obstacles during the movement, if the execution probability of the gripper is ignored.

When the position of the object sensed by the sensor is $\boldsymbol{X}$, and the transformation matrix from the base of the arm to the camera is $\boldsymbol{A}$, the position of the target based on the arm's base is $\boldsymbol{Y}_{\text{sense}} = \boldsymbol{AX}$. The

Algorithm 1 Decision-making loop

```
Require: world_states, rules, goals
 1: replan ← true
 2: while replan = true do
 3:     states, actions ← solver(world_states, rules, goals)
 4:     replan ← false
 5:     if states = ∅ and actions = ∅ then
 6:         fail // no result
 7:     end if
 8:     if states ≠ ∅ and actions = ∅ then
 9:         task is finished
10:     else
11:         $i \leftarrow 0$
12:         while $i$ <len(actions) do
13:             belief_states ← states[$i$]
14:             update percept_states from perception
15:             if conflict(belief_states, percept_states) then
16:                 world_states ← update(percept_states)
17:                 replan ← true
18:                 break while
19:             end if
20:             result ← execute_action(actions[$i$])
21:             if result = true then
22:                 world_states ← update(states[$i+1$])
23:                 $i \leftarrow i + 1$
24:             else
25:                 replan ← true
26:                 break while
27:             end if
28:         end while
29:         if replan = false then
30:             task is finished
31:         end if
32:     end if
33: end while
```

error caused by slight looseness and misalignment of hardware is $\boldsymbol{E}$, and the real position is $\boldsymbol{Y}_{\text{real}} = \boldsymbol{AEX}$. Then the error matrix of the object sensed by the sensor is

$$\text{error\_matrix} = \boldsymbol{Y}_{\text{sense}}^{-1}\boldsymbol{Y}_{\text{real}} = \boldsymbol{X}^{-1}\boldsymbol{EX}.$$

If error_matrix is represented by rotation matrix $\boldsymbol{R}_M$ and position matrix $\boldsymbol{T}_M$, the error related to the pose of $\boldsymbol{X}$ is

$$\text{error} = ||\boldsymbol{T}_M||_2 = ||\boldsymbol{R}_X^{\text{T}}(\boldsymbol{R}_X - \boldsymbol{I})\boldsymbol{T}_X + \boldsymbol{T}_E||_2,$$

which means the error is related to the errors in misalignment, especially in the error of rotation and the distance between the camera and the sensed object. However, when the robot senses the object and the gripper at the same time, where the coordinate position between the gripper and the object is $\Delta\boldsymbol{X} = \boldsymbol{X}_{\text{object}}^{-1}\boldsymbol{X}_{\text{gripper}}$, the error matrix of the coordinate position based on the arm's base is

$$\text{related\_error\_matrix} = \Delta\boldsymbol{X},$$

which means the error is the distance between the gripper and the object, $d(\text{gripper}, \text{object})$. While the operation is proceeding, related_error is always smaller than the error. It leads to the result that the perception of the coordinate position between the gripper and the object is more accurate than the absolute position of the object. Therefore, this feature can be used to reduce errors in the operating process. What is more, when the gripper is very close to the object and the robot's arm runs with local stability, which means the joints remain stable in a localized range where there is no applied load or a sudden change in speed, the related_error is independent of errors in misalignment and is too small to disturb the operation.

6.1 Method

The manipulation method is shown in Algorithm 2. When performing an object operation, KeJia first uses RRT (Kuffner and LaValle, 2000) to plan the trajectory and check if the path is valid. If it is, this trajectory will be executed; otherwise, the location of the failure is detected. If the location occurs near the target point, the robot executes the partial trajectory to the blocking location and then enters the end-to-end layer; otherwise, KeJia needs to sense the distance between obstacles and the gripper to reduce the error of perception. Then a new error model is updated and a replan occurs. If the error cannot be reduced to make KeJia have an effective motion planning solution or enter the end-to-end layer, the operation fails.

Algorithm 2 Manipulation

```
Require: obstacles, robot_model, retry_limit
 1: retry ← 0
 2: while retry < retry_limit do
 3:     trajectories ← planning(obstacles, robot_model)
 4:     if trajectories ≠ ∅ then
 5:         $i$_model ← inflact(robot_model, error)
 6:         if isValid(obstacles, $i$_model) then
 7:             execute trajectories
 8:         else
 9:             execute trajectories to blocking location $p$
10:             if inRange($p$) then
11:                 enter the end-to-end control loop
12:             else
13:                 percept again and reduce error
14:                 error ← new_error
15:                 retry ← retry + 1
16:             end if
17:         end if
18:     else
19:         fail
20:         break while
21:     end if
22: end while
```

6.2 End-to-end layer

In the end-to-end layer, a CNN-structured network $g(I_t, a_{t-1})$ is used to evaluate the moving direction of the robot's arm from calibrated images in the micro camera. I_t is the calibrated image collected from the micro camera, which is attached to KeJia's gripper, and a_{t-1} is the action which the robot chose in the last controlling loop. This network is processed in every control loop. Details are shown in Algorithm 3.

Algorithm 3 End-to-end control loop

```
 1: $a_{t-1} \leftarrow [1,0,0,0,0]$ // action: move forward
 2: loop
 3:     obtain current image $I_t$
 4:     $a_t \leftarrow g(I_t, a_{t-1})$
 5:     action ← max($a_t$)
 6:     execute action
 7:     $a_t \leftarrow a_{t-1}$
 8:     if collision is detected then
 9:         stop
10:         break loop
11:     end if
12: end loop
13: if the target pose is reached then
14:     close/open gripper
15: else
16:     return fail
17: end if
```

The end-to-end layer has three assumptions: (1) During the gripping process, both the object which is being manipulated and the gripper are in the field of view; (2) After the planning level is executed, a certain error exists between the gripper and the target items; (3) The robot's arm runs with local stability, which means the joints remain stable in a localized range where there is no applied load or a sudden change in speed.

To ensure the real-time performance of closed-loop operation, the network should not be too complicated. An AlexNet-variant network is used. The architecture of our network is shown in Fig. 3. The output of the network is a 5D vector: [go, up, down, left, right]. The loss function of the network is estimated using cross entropy. The action which has the highest score in the output is chosen as the moving direction to be executed in the current controlling loop. When the gripper successfully presses the button, the electric current of each joint's motor will abruptly change. It is considered that the manipulation phase is completed.

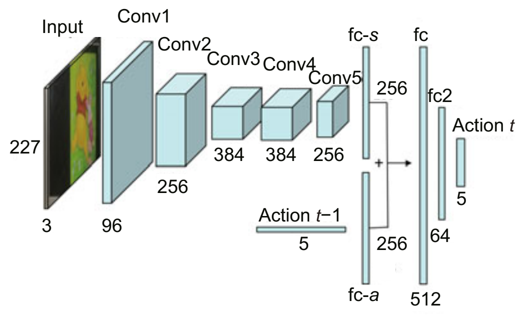


Fig. 3 Network structure

6.3 Data collection

The data collection process is divided into two parts: one for data about the influence of the robot's actions on the environment, and the other for data about the correspondence between the environmental state and the sensor image. Data for these two parts can be separately collected.

First, we use a reality-simulation-unified method to collect data. In the real world, we let the robot make movements on the empty ground, while using motion capture devices (Corazza et al., 2006) to capture the motion of the robot. In the simulator, we maintain a virtual target object and a virtual gripper with the same shape as KeJia's end-effector and the same motion of KeJia captured by motion capture devices. In this way, the robot can easily obtain enough motion data without objects.

Next, we simply capture images from the micro camera on the gripper, and the positional relationship between the object and the gripper is captured by the motion capture device at the same time. Because of the extremely small range of motion (assumption (3) in Section 6.2), it is easy to collect enough image data.

Finally, the images are combined with the corresponding action positions to form training data.

7 Experiment

In this section, our experiment can be divided into three parts: (1) We evaluated our end-to-end layer's performance by training networks to grasp objects which are rendered in the simulator, and evaluated network's performance of grasp quality in the simulator. (2) We evaluated our manipulation method in the real world and compared it with three other different methods. For each method, 45 attempts were allowed to press the button of an oven. (3) To evaluate the integration of the whole system of KeJia, a case study was produced when cooking popcorn using an oven.

7.1 Simulation of the end-to-end layer

We trained the end-to-end layer and evaluated the performance of the Gazebo simulator (Koenig and Howard, 2004). To evaluate the three assumptions made in Section 6.2, we tested the end-to-end layer with the following six factors: (1) numbers of samples; (2) start position; (3) randomization of the background and light; (4) changes of the camera's positions; (5) whether the gripper is in the field of view; (6) use of the pre-trained weight.

Two types of targets were employed in the experiment: one is a large object (such as an oven) which can cover the field of view, and the other is a small object (such as a bottle or a pen). The difference of these two types of objects is that the background of the environment can be seen through a bottle or a pen, but not through a oven. The learning rate was set as 1×10^{-6} for fine-tuning.

The grasp position was set at the object's center. According to the distance between the final position of the gripper and the grasp position, we divided the performance of grasping into five levels: perfect ($<$ 2 mm), good (2–3 mm), not bad (3–4 mm), pass (4–5 mm), and fail ($>$5 mm). We made 200 attempts for each test.

The range of captured training data was 5 cm around the target's center.

Fig. 4 shows the relationship between the number of samples and grasp quality. The start positions in this experiment were random locations within 5 cm around the target. We achieved a high grasp feasibility with 3000 images, which means it takes only a few minutes to collect enough data.

Fig. 5 shows that the network has a different performance at different start positions. Grasp quality can be guaranteed in the range of training data. A pre-trained model can achieve satisfactory results in a wider range than the model without pre-trained

Table 1 **Grasp feasibility of different methods with 200 attempts**

Method	Success times				Total success times
	$d \in [0,2)$ mm	$d \in [2,3)$ mm	$d \in [3,4)$ mm	$d \in [4,5)$ mm	
Full method	112	3	63	22	200
Without background randomization	101	0	0	92	193
Without gripper in the field of view	0	0	1	41	42
Without camera randomization	0	0	88	86	174

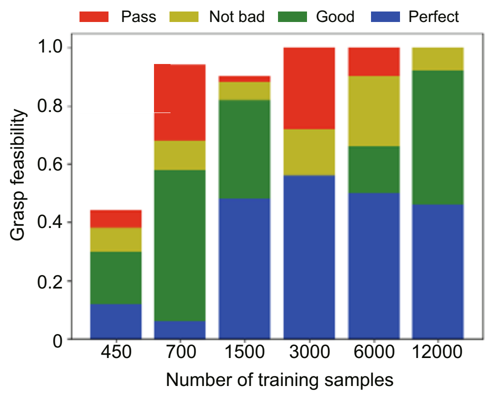


Fig. 4 Relationship between the number of samples and grasp quality

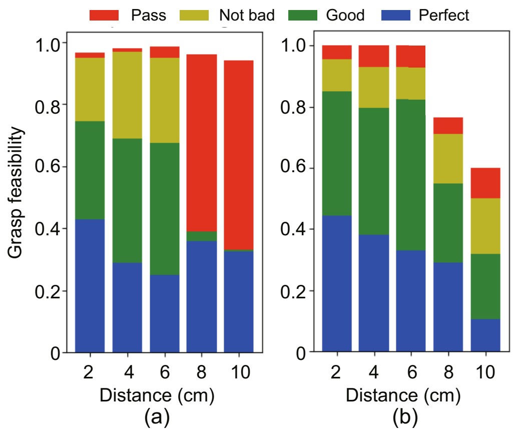


Fig. 5 Relationship between grasp quality and the start position with (a) or without (b) pre-trained weights

weights.

Table 1 shows the importance of randomization of the background and light, changes of camera's position, and whether the gripper is in the field of view by removing these factors when capturing data. Start positions in this experiment were random locations within 5 cm around the target, and the camera's position was changed when we evaluated the performance of grasping. We found that randomization was needed especially in the camera's position. The randomization of background slightly improved the effect. It stemmed from the fact that most of the view of the camera was covered by the target. In addition, the gripper needed to be seen in the view of the camera, and the network did not converge, leading to a bad feasibility.

7.2 Manipulation in the real world

In this experiment, we let KeJia try to press the button of the oven. KeJia was asked to navigate to a random place 2 m around the oven and started its attempts. If KeJia successfully pressed the button, the GUI of the screen on the microwave oven will change, allowing us to autonomously evaluate the results of the operation by detecting if the screen had changed after the robot finished its action.

Experimental results showed that the random error of execution was ±3 cm, while the button's diameter was only 1 cm. If KeJia performed the action of pressing the button under the influence of various errors, depending on the visual result, the gripper would randomly reach a position within 3 cm of the button's midpoint. It had a great probability of failing in the action of pressing the button as the button was too small.

We evaluated the performances of four different methods, including the classical controlling method and the end-to-end method. The first method is an open-loop controlling method, recognizing the target using a depth camera and controlling its arm to the calculated position. The next method is a closed-loop controlling method of hand-eye calibrated (Cui et al., 2018), attaching a marker to the gripper. The third method is a full end-to-end method (Levine et al., 2018). Levine et al. (2018) let the robot autonomously find the best way, which is different from our work where the learning is supervised. As a result, instead of letting the robot explore by itself, we simply generated the actions labels, collected data under human control, and used the network structure proposed by Levine et al. (2018). The last one is our method, training the end-to-end layer's network using the full method with pretrained weights, and the learning rate was set as 1×10^{-6} for fine-tuning. All methods were limited to collecting data in 1 min.

For each method, 45 attempts were allowed. In the first 15 attempts, we let the robot undisturbedly press the button. In the next 15 attempts, we manually pushed the robot's arm away from the target to find out whether it can recover by itself. It must be mentioned that the robot's arm has gear clearance. In the last 15 attempts, we changed the orientation of the micro camera on the gripper and the depth camera on the robot by several degrees, and evaluated whether it can overcome this calibration error. The results are shown in Table 2.

Table 2 Success times in attempts of pressing button

Method	Success times		
	First 15 attempts	Second 15 attempts	Third 15 attempts
Open-loop	15	3	7
Closed-loop	14	15	12
End-to-end	–	–	–
Ours	15	14	15

Our method succeeded in 44 attempts. The attempt that failed happened in the second 15 attempts, during which we pushed the robot's arm when the robot moved too close to the oven, and thus the robot did not have enough space to adjust its gripper's position. The open-loop method did well in the first 15 attempts but badly in the other 30 attempts. In the closed-loop controlling method, the robot could recover when errors occurred, but it could not recover when it needs to calibrate itself. In addition, a special marker was needed to recognize the robot's position. The failed attempts all occurred in the situation where the marker was covered by the robot's body. As for the full end-to-end method, our dataset was too small to grasp objects.

7.3 Case study

In this case study, KeJia was constantly affected by the following uncertainties: errors in perception (e.g., errors in detecting objects' positions), changes in the environment (e.g., human assistance and interference), and errors in hardware (e.g., errors in the robot arm model and in the transform relationship between the camera and arm's base). In the experiment, we used iclingo (Gebser et al., 2008) as our solver. For each object (i.e., microwave oven, cup, and button) used in the demonstration, we spent 1 min collecting data. For the task, KeJia needed to put a cup of corn into the microwave oven and then to press the button on the oven to make popcorn.

Our demo video (http://ai.ustc.edu.cn/media/kejia_oven.mp4) shows how KeJia dealt with failures and errors occurring in the task of cooking a cup of popcorn using a microwave oven (Fig. 6).

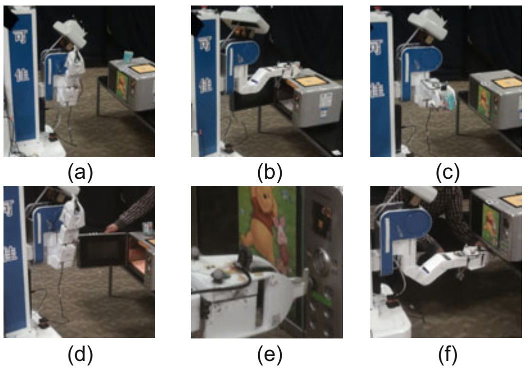


Fig. 6 Scenarios in demonstration: (a) scenario 1; (b) scenario 2; (c) scenario 3; (d) scenario 4; (e) scenario 5; (f) scenario 6

Scenario 1 The operator wanted to heat popcorn. While the robot was preparing for the first action occurs(open(1),1), it noticed that the oven's door was closed, corn was on the top of the oven, and its gripper was empty.
Scenario 2 For the action occurs(grasp(2),2), KeJia planed its motion using the RRT method and noticed that it could execute actions.
Scenario 3 Before Kejia was going to put the cup into the oven, it noticed that the door was closed and its gripper was occupied. Therefore, KeJia replanned its action. It decided to put the cup somewhere and then opened the door again using its empty gripper.
Scenario 4 An extra change in state was detected. The operator opened the oven's door and adjusted the position of the oven. KeJia replanned its action again.
Scenario 5 For the action occurs(grasp(2),1), KeJia planned its motion using the RRT method and noticed that it could execute actions. Then it continued its actions in the end-to-end layer using cameras on the gripper and successfully pressed the heat button.
Scenario 6 KeJia failed in the action occurs(open

(1),4) because of the operator's disturbance. It is noticed that it failed to open the door and tried the action once again.

8 Conclusions

In this paper, the frame of service robot KeJia which is aimed for home scenarios has been introduced. Our work has focused on integrated decision making and error-tolerant manipulation. We have developed an intelligent service robot system which can flexibly handle a dynamic environment and tolerate errors during manipulation.

Using the solver based on the ASP and continuous observation, KeJia can adapt itself to some unexpected changes in the environment. This method does not need to program each possible situation by hand to deal with all kinds of possible states, but needs to describe only the condition and influence of the action after the realization of functional modules.

We have also proposed a novel hierarchical method which combines motion planning with neural network prediction to enable robots to use low-precision equipment to complete high-precision actions. Instead of collecting a large amount of data for a long time, only a simple one-minute data collection is needed for each operating object, which greatly reduces the cost of data collection in robotic applications. We hope that our work will provide some experience to similar robots.

References

Blackburn P, Bos J, 2005. Representation and inference for natural language: a first course in computational semantics. *Comput Ling*, 32(2):283-286.

Bohren J, Cousins S, 2010. The SMACH high-level executive [ROS news]. *IEEE Robot Automat*, 17(4):18-20. https://doi.org/10.1109/MRA.2010.938836

Chen K, Lu DC, Chen YF, et al., 2014. The intelligent techniques in robot *KeJia*—the champion of RoboCup@Home 2014. *LNCS*, 8992:130-141. https://doi.org/10.1007/978-3-319-18615-3_11

Chen K, Yang FK, Chen XP, 2016. Planning with task-oriented knowledge acquisition for a service robot. Proc 25th Int Joint Conf on Artificial Intelligence, p.812-818.

Chen XP, Ji JM, Jiang JQ, et al., 2010. Developing high-level cognitive functions for service robots. Proc 9th Int Conf on Autonomous Agents and Multiagent Systems, p.989-996.

Chen YF, Shuai W, Chen XP, 2015. A probabilistic, variable-resolution and effective quadtree representation for mapping of large environments. Int Conf on Advanced Robotics, p.605-610. https://doi.org/10.1109/ICAR.2015.7251518

Corazza S, Müendermann L, Chaudhari AM, et al., 2006. A markerless motion capture system to study musculoskeletal biomechanics: visual hull and simulated annealing approach. *Ann Biomed Eng*, 34(6):1019-1029. https://doi.org/10.1007/s10439-006-9122-8

Cui GW, Chen GD, Zhang ZK, et al., 2018. A flexible grasping policy based on simple robot-camera calibration and pose repeatability of arm. Int Conf on Intelligent Robotics and Applications, p.89-99. https://doi.org/10.1007/978-3-319-97589-4_8

Erdem E, Aker E, Patoglu V, 2012. Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *Intell Serv Robot*, 5(4):275-291. https://doi.org/10.1007/s11370-012-0119-x

Gebser M, Kaminski R, Kaufmann B, et al., 2008. Engineering an incremental ASP solver. *LNCS*, 5366:190-205. https://doi.org/10.1007/978-3-540-89982-2_23

Grisetti G, Stachniss C, Burgard W, 2007. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Trans Robot*, 23(1):34-46. https://doi.org/10.1109/TRO.2006.889486

Klein D, Manning CD, 2003. Accurate unlexicalized parsing. Proc 41st Annual Meeting on Association for Computational Linguistics, p.423-430. https://doi.org/10.3115/1075096.1075150

Koenig N, Howard A, 2004. Design and use paradigms for Gazebo, an open-source multi-robot simulator. IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.2149-2154. https://doi.org/10.1109/IROS.2004.1389727

Krizhevsky A, Sutskever I, Hinton GE, 2012. ImageNet classification with deep convolutional neural networks. Proc 25th Int Conf on Neural Information Processing Systems, p.1097-1105.

Kuffner JJ, LaValle SM, 2000. RRT-connect: an efficient approach to single-query path planning. Proc IEEE Int Conf on Robotics and Automation, p.995-1001. https://doi.org/10.1109/ROBOT.2000.844730

Levine S, Pastor P, Krizhevsky A, et al., 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int J Robot Res*, 37(4-5):421-436. https://doi.org/10.1177/0278364917710318

Lifschitz V, 2008. What is answer set programming? Proc 23rd AAAI Conf on Artificial Intelligence, p.1594-1597.

Mahler J, Liang J, Niyaz S, et al., 2017. Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. https://arxiv.org/abs/1703.09312

Murray RM, 2017. A Mathematical Introduction to Robotic Manipulation. CRC Press, London, UK.

Pinto L, Gupta A, 2016. Supersizing self-supervision: learning to grasp from 50K tries and 700 robot hours. IEEE Int Conf on Robotics and Automation, p.3406-3413. https://doi.org/10.1109/ICRA.2016.7487517

Popov I, Heess N, Lillicrap T, et al., 2017. Data-efficient deep reinforcement learning for dexterous manipulation. https://arxiv.org/abs/1704.03073

Quigley M, Conley K, Gerkey B, et al., 2009. ROS: an open-source robot operating system. Proc ICRA Workshop on Open Source Software, p.1-6.

Rusu RB, Cousins S, 2011. 3D is here: Point Cloud Library (PCL). IEEE Int Conf on Robotics and Automation, p.1-4. https://doi.org/10.1109/ICRA.2011.5980567

Sakagami Y, Watanabe R, Aoyama C, et al., 2002. The intelligent ASIMO: system overview and integration. IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.2478-2483. https://doi.org/10.1109/IRDS.2002.1041641

Ulrich I, Borenstein J, 1998. VFH+: reliable obstacle avoidance for fast mobile robots. Proc IEEE Int Conf on Robotics and Automation, p.1572-1577. https://doi.org/10.1109/ROBOT.1998.677362

Vannoy J, Xiao J, 2008. Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Trans Robot*, 24(5):1199-1212. https://doi.org/10.1109/TRO.2008.2003277

Wisspeintner T, van der Zant T, Iocchi L, et al., 2009. RoboCup@Home: scientific competition and benchmarking for domestic service robots. *Interact Stud*, 10(3):392-426. https://doi.org/10.1075/is.10.3.06wis