International Journal of Humanoid Robotics

POURING SKILLS WITH PLANNING AND LEARNING MODELED FROM HUMAN DEMONSTRATIONS

AKIHIKO YAMAGUCHI

The Robotics Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh PA 15213-3890, United States,
Graduate School of Information Science, Nara Institute of Science and Technology,
8916-5, Takayama, Ikoma, Nara 630-0192, Japan,
info@akihikoy.net

CHRISTOPHER G. ATKESON

The Robotics Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh PA 15213-3890, United States,
cga@cs.cmu.edu

TSUKASA OGASAWARA

Graduate School of Information Science, Nara Institute of Science and Technology,
8916-5, Takayama, Ikoma, Nara 630-0192, Japan,
ogasawar@is.naist.jp

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

We explore how to represent, plan, and learn robot pouring. This is a case study of a complex task that has many variations, and involves manipulating non-rigid materials such as liquids and granular substances. Variations of pouring we consider are the type of pouring (such as pouring into a glass or spreading a sauce on an object), material, container shapes, initial poses of containers, and target amounts. The robot learns to select appropriate behaviors from a library of skills, such as tipping, shaking, and tapping to pour a range of materials from a variety of containers. The robot also learns to select behavioral parameters. Planning methods are used to adapt skills for some variations such as initial poses of containers. We show using simulation and experiments on a PR2 robot that our pouring behavior model is able to plan and learn to handle a wide variety of pouring tasks. This case study is a step towards enabling humanoid robots to perform tasks of daily living.

Keywords: Pouring; learning from demonstration; learning from practice; planning.

1. Introduction

The goal of this work is to explore how to represent, plan, and learn complex tasks that have many variations. We want to enable robots to go beyond manipulating rigid bodies. Pouring, a form of manipulating liquids and granular materials, is

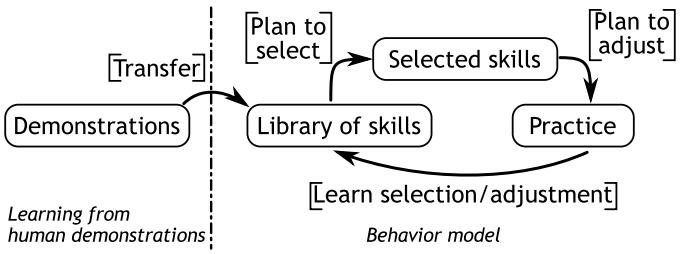


Fig. 1. Conceptual illustration of our learning from demonstration scheme. Left: learning from human demonstrations, Right: behavior generation and refinement.

an example of such a task. Materials we pour range from water to more viscous liquids such as shampoo. In making a pizza, a number of pouring skills are used: pouring cheese from a bag, pouring vegetables from a bowl, pouring tomato sauce with shaking or squeezing a bottle, pouring seasonings, and so on. Pouring can involve tipping, shaking, and tapping a container. Pouring has different variations involving many materials, container shapes, contexts, initial poses of containers, target amounts, and obstacles. In order to handle these variations, we humans use many strategies or *skills*.

We want robots to help us in daily activities involving non-rigid materials. Examples include making pancakes [1], folding towels [2], and baking cookies [3]. A common challenge is handling variations of these tasks. Tools that have been developed for this challenge include search-based planning algorithms [4], reinforcement learning methods [1], and supervised learning methods. In our experience these tools will solve a specific version of a task and have limited generalization ability. We need a methodology that combines planning and learning and has wider generalization.

In this case study we informally model robot behaviors on human examples, exploring a learning from demonstration approach [5]. Phases of pouring include a preparation process (grasping a source container and moving the container to the receiving container's position), controlling the material flow, and a post pouring process (putting the source container at a final position). The most difficult thing to model is flow control, since it is a manipulation of complex materials, such as liquid and granular materials. Humans use skills such as shaking for flow control. We think humans learn those skills, including how to select a particular strategy and adjusting behavioral parameters, such as shaking angle. In addition, humans can adapt their skills to a new situation, i.e. a new combination of containers and material; as well as improving their performance through learning from practice.

We consider a behavior model consisting of a library of skills, and planning and learning methods. Our architecture is illustrated in Fig. 1. Though our goal is automating the entire architecture, in this paper we focus on the right part, and do the left part manually; i.e. we model the skills manually. The skills are modeled with a well-known representation, finite state machines. Finite state machines represent

both a behavioral structure (grasping, moving container, etc.) and feedback control (e.g. tipping until a target amount is achieved). Planning methods are introduced to obtain some situation specific parameters such as grasp parameters, pouring locations, and so on. Learning from practice methods are introduced for skill selection, for parameter adjustment, and for improving plan quality. Learning from practice for skill selection is important for generalization of flow control since it is difficult to model liquid or granular material dynamics.

We implemented the pouring behavior model on a PR2 robot and conducted both simulation and robot experiments. We verified the generalization ability of the model in terms of materials, container shapes, contexts, container locations, and target amounts.

The most important finding of this case study is that in order to model a behavior with wide generalization, it is a practical solution to store small skills (e.g. tipping, shaking, grasping) in a library, and combine them for an entire task, using planning and learning methods for selection and adjustment. Though we cannot say this approach is the best, its practicality is verified by the robot experiments.

Section 2 provides an overview of our pouring model, and we compare it with related works in Section 3. In Section 4, we discuss human pouring behavior. Section 5 describes our modeling of pouring behavior. The details of learning and planning methods are described in Section 6 and 7. Section 8 describes the experiments, and we conclude in Section 9.

2. Overview of the Pouring Model

In this section, we describe the pouring variations considered, our assumptions, an overview of our pouring behavior, and how generalization is achieved.

2.1. *Pouring Variations*

We consider following pouring variations:
Material type: We explore the effects of the type of material to be poured, such as water, coke, sugar, coffee powder, tomato sauce, and shampoo. In the experiments we use dried granular materials instead of liquid in order to avoid damage to the robot.
Container shape: We use various shapes of source containers such as a cup, a bottle with a small mouth, and a can.
Context: We consider multiple pouring contexts, including pouring a cup of water during serving it to a user, and spreading sauce on a surface during making a pizza. Context determines the goal of pouring.
Initial poses of containers: Initial positions and orientations of containers are varied. Considering an orientation is important especially when pouring from a coke can-like container whose mouth position is not at the center of the container top.
Target amount: Target amount of the poured material, such as 90% of the height of a receiving container, is also varied.

2.2. *Assumptions*

We assume the following about the pouring task:

- The robot uses its left hand (if it has two arms).
- Each container is already opened (if it has a lid or top).
- A library of container models is given, where each model consists of a cylinder for a graspable part and a polygon for the mouth. For the current work, labels of container types and material types are given. Future work will obtain these from perception.
- The amount of poured material into a receiving container is observable.
- The initial poses of the containers are observable.

2.3. *Overview of the Pouring Behavior Model*

The resulting behavior model has the following elements:
Finite state machines: Each skill, such as pouring by tipping, shaking, and tapping is modeled with finite state machines that have two roles: feedback control and procedural execution. For example in pouring by tipping, the state machine works as a feedback controller where a current state such as a poured amount is observed, and a control signal is computed accordingly. A general flow controller is modeled as a hierarchical state machine using these skills. Each skill may have some parameters for selection and adjustment.
Decomposing pouring behavior: The entire pouring behavior is decomposed into several sub-skills and modeled by state machines, such as grasping and moving a container. Such a decomposition is possible because the procedure of sub skills is usually the same (moving an arm $\rightarrow$ grasping a container $\rightarrow$ moving the container $\rightarrow$ flow control $\rightarrow \ldots$). This decomposition is also compatible with a memory-based approach to flow control, since the memorized skills are considered as decomposed skills.
Planning methods: Similarly, rather than planning many steps of the pouring behavior at once, we separate it into several small planning problems, such as grasping, pouring location, and path. The benefit is that we can reduce the computational cost of planning.
Learning from practice: We consider two types of learning methods: (1) direct policy learning for selection (e.g. skill selection) and for adjustment (e.g. shaking angle), and (2) learning to improve planning where we update the evaluation function used in planning, rather than the planned policy.

Unifying these elements, we can achieve a pouring behavior model with wide generalization in terms of the variations mentioned above. Table 1 summarizes how much each element contributes to each generalization ability. Since some combinations of material types and container shapes require different flow control skills, state machines representing skills and skill selection learning are essential. Learning adjustment methods may also improve performance. Planning methods are necessary

Table 1. How much each element contributes to generalization.

Method	Material type	Container shape	Context	Initial poses	Target amount
State machines for skills	**	**	**	*	**
Planning methods		**		**	
Learning for planning methods		*		*	
Learning for selection	**	**			
Learning for adjustment	*	*			

** : Plays an essential roll.
* : Plays an important roll in improving performance.

since container shape affects the grasp parameters, hand path, etc. The difference of contexts is handled by state machines. The variation of initial poses is mainly handled by planning methods, but a state machine may increase the success rate in different initial poses; for example, the gripper position is improved by visual feedback control represented with a state machine. The difference of target amounts is handled by state machines for feedback control of the target amount observation. The learning method to improve planning improves plan quality.

3. Related Works

3.1. *Pouring Robots*

There are several attempts to enable robots to learn pouring from human demonstrations [6,7,8,9,10,11,12]. However, they typically focus on only a part of the entire pouring problem. For example, Tamosiunaite *et al.* proposed a method to optimize the source container trajectory and its goal position for a pouring task [8]. This method may be able to generalize at a certain level, but we cannot expect generalization of source container shapes and material types. Rozo *et al.* proposed a method to teach a robot to pour using force information, and models the human demonstration with a parametric hidden Markov model (HMM) [11]. Using HMM-like models is useful to encode a human demonstration automatically, but it is just a part of the entire pouring problem. Pastor *et al.* made a robot learn dynamic movement primitives (DMPs) for pouring behavior from human demonstrations [7]. Their method is limited to generalization of goal positions. Kroemer *et al.* proposed a method for robots to learn behaviors like pouring from human demonstrations [10]. Their method enables the robots to learn behaviors for novel objects through trial and error. Brandl *et al.* proposed to use warped parameters for automatic generalization of behaviors between differently shaped objects [12]. Those methods can generalize in terms of container shapes, but it may be difficult to generalize behaviors for material types where different skills are used.

Therefore, although our approach relies on manual skill implementation, the obtained pouring behavior has a large generalization ability in terms of several variations of pouring as discussed in Section 2. Generalizing behaviors for material

types is difficult; our solution is using a library of skills like tipping and shaking, and selecting automatically an appropriate skill.

3.2. *Planning and Learning Methods*

The learning from practice of this paper is close to the "learning from observations and practice using behavioral primitives" framework proposed by Bentivegna [13]. Another similar approach is found in the "learning parameterized skills" framework [14].

Kaelbling *et al.* developed a hierarchical planning method where symbolic-level task planning and motion planning are integrated [15]. They are exploring variations of planning that have a hierarchical structure, while our planning variations are independent of one another.

There is some work on improving planning through learning from practice. For example, Zucker *et al.* proposed Reinforcement Planning where a cost function for the planner is improved through actual execution [16]. In this paper, we present a similar but simpler solution, but it is also possible to use their algorithms.

Bollini *et al.* proposed the BakeBot which can analyze a recipe, plan a sequence of actions where motion primitives, such as mix, pour, and bake, are used, and execute the sequence [3]. They experimentally demonstrated that their system could successfully follow two different recipes with a real robot. Though their direction and ours are similar, this paper includes learning methods, combines them with planning, and considers a larger skill library.

4. Discussions of Human Pouring

We discuss the features of human pouring demonstrations. First, we focus on flow control, and then we discuss the entire pouring process.

4.1. *Flow Control*

Pouring by Tipping

We use the setup shown in Fig. 2 to track human demonstrations. A human subject pours from a source container (source) to a receiver container (receiver) where the orientation of the source and the amount of material in the receiver are measured by RGB cameras. The material in the source is dried peas which behave like water, but are more convenient for measuring the amount and for robot experiments especially when spills occur. The human subject pours the material to a target amount of 0.5 which is half of the receiving container height.

Fig. 3 shows a human demonstration of pouring where the amount in the receiving container and the orientation are plotted. We can see that there are three phases in pouring. Phase 1: rotating the source container quickly until flow is observed. Phase 2: after flow starts, the human rotates the source container slowly

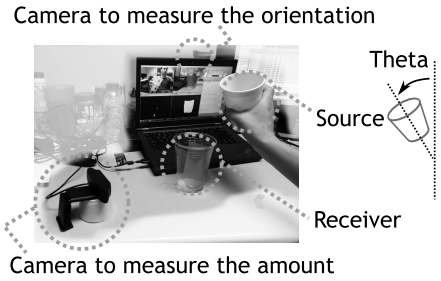


Fig. 2. Setup to measure a human demonstration.

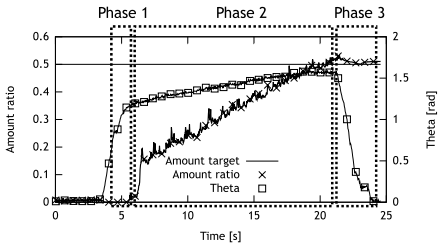


Fig. 3. Result of a human demonstration. The amount trajectory and the orientation (Theta) trajectory are plotted.

(a) Shaking A.

(b) Shaking B.

(c) Tapping.

Fig. 4. Human demonstrations of shaking and tapping.

until the amount reaches the desired target. We found that once flow starts, it continues without rotating the container. Phase 3: after reaching the target amount, the human moves the source container to the final pose.

From the human demonstrations, we found that during pouring, the mouth edge of the source container did not move much, while the grasping point moved more. Modeling the movement of a point on the mouth edge is easier than modeling the movement of the gripper.

Pouring by Shaking and Tapping

Humans use different strategies if pouring by tipping does not work. For example when the material is jamming inside the container or pouring slowly, humans shake and/or squeeze the container. When humans want to pour a small amount (especially of granular materials), they may tap the container. In this paper, we consider shaking and tapping skills.

A shaking skill is used for jammed material or for viscous liquids. Through human demonstrations, we found that there are some variations in shaking behaviors;

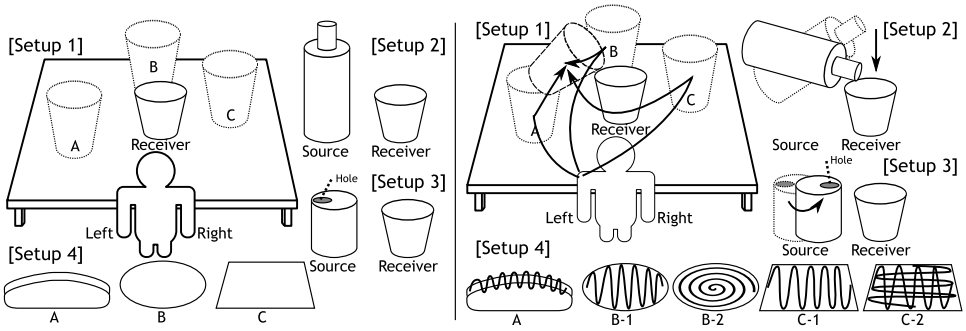


Fig. 5. Examples of pouring behaviors of humans. Left: situations, Right: human behaviors.

for example, shaking vertically, shaking at an angle, shaking linearly, and shaking rotationally. Fig. 4(a) and 4(b) show the examples of shaking motions demonstrated by a human, where the mouth of a bottle is modified to jam material.

Tapping is used to pour material accurately; for example, pouring coffee powder. Fig. 4(c) shows a human demonstration of tapping, where the human is holding a container with the left hand, and tapping the container with a right finger.

4.2. *Pouring Variations*

Next, we discuss different variations of pouring. Here we conducted an informal user study where we asked four people to perform pouring in four setups as illustrated in Fig. 5, took videos, and analyzed them manually. We asked each subject to use only the left hand in order to decrease the modeling complexity.

Setup 1

A source container is put around a receiving container at various locations (A, B, C). In every location, the subjects moved the left hand from the initial pose to a grasping pose of the source container, and moved the source container toward the pouring location on the receiving container. In each movement, the subjects avoided collisions between their body (including grasped objects) and objects in the environment.

An interesting thing we found is that all subjects poured from the left side of the receiving container in the A and C locations, and in the B location, two out of four subjects poured from the left side and two subjects poured from between the rear and the left side. We believe this maintains controllability during actual pouring. If a human pours from right side of the receiving container using the left hand, the posture during pouring would be unfamiliar and uncomfortable to the human.

Setup 2

The height of the source container is taller than that of the receiving container. In Section 4.1, it was found that the pouring position on the receiving container does not move much during actual pouring. But in Setup 2, the subjects started pouring at a certain height, then moved the mouth of the source container downward during pouring. We believe this occurs because one cannot start pouring with the lower height because the bottom of the source container collides with the table. Moving the pouring position downward avoids spilling.

Setup 3

The mouth (hole) of the source container is not located at the center of the container, as in a coke can, and initially the position of the hole is on the opposite side of the receiving container. Since the subjects were asked to use only their left hands, each subject manipulated the source container to change the hole position to the right side, then started pouring. Two out of four subjects manipulated the container inside their hands, and two subjects rotated the container by putting it on the table. We believe the purpose of such *re-grasping* is to change the infeasible pouring setup to a feasible one.

Setup 4

We explored spreading materials onto several types of receivers that have different shapes (e.g. hot dog bread, pizza, and square bread). We asked the subjects to illustrate patterns on a paper with a pen by imagining spreading sauce on breads from a thin tip container. The subjects adapted the spreading trajectories to each receiver's shape with various patterns, for example B-1 and B-2, and C-1 and C-2.

5. Modeling Pouring

Based on the human demonstrations, we discuss how to model pouring behavior for a robot. We model the behavior with finite state machines that work as both a control procedure (e.g. tipping until the target amount is poured) and a structural procedure (e.g. moving hand, grasping, moving container, ...). In this section, first we briefly describe our finite state machines, then we model flow control. Finally we model the entire pouring behavior.

5.1. *Finite State Machines*

Pouring consists of several steps: reaching to a source container, grasping it, moving the source container to a receiving container, actual pouring (flow control), moving the source container to a position, releasing it, and moving the hand to a final position. We use a finite state machine to represent the behavior. Since there are

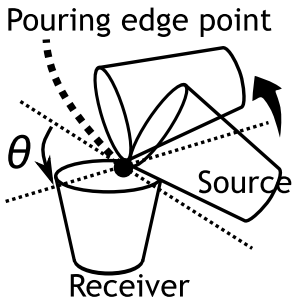


Fig. 6. Illustration of tipping model.

high-level procedures, such as a preparation and flow control, and detailed procedures, such as shaking, we use hierarchical finite state machines.

In addition to this, we introduce parallel state machines. For example in Setup 2 of Section 4.2, the behavior representation can be simplified by using a parallel model. One state machine controls the flow, and another state machine controls the height of the source container's mouth.

5.2. *Flow Control Skills*

A core skill in pouring is flow control where the robot handles non-rigid materials. We model flow control based on human demonstrations, where we assume that the robot starts to pour when the robot is grasping a source container and holding it near the receiving container.

5.2.1. *Tipping*

According to the human demonstrations, humans use different strategies based on the relative height of the source container and the receiving container. Since a smaller source container case is simpler, we consider that case first. The other case is considered in the general flow control section (5.2.4). We can assume that during flow control, the edge point of the source takes a constant value, and only the orientation changes to control the flow, as illustrated in Fig. 6. The source container moves mostly in a 2-dimensional plane, so the orientation is modeled by a 1-dimensional variable, θ.

As we mentioned in Section 4.1, there are three phases in the human demonstration. We model this behavior using a finite state machine. When no flow is observed, the robot increases θ (Phase 1). If flow is observed, the robot slows down the movement (Phase 2). If the target amount is achieved, the robot moves θ to the final value (Phase 3). In some demonstrations, we found that if the material starts flowing, it continues to flow without increasing θ. Thus, in Phase 2, we increase θ only when flow is not observed, and keep the same value when flow is observed.

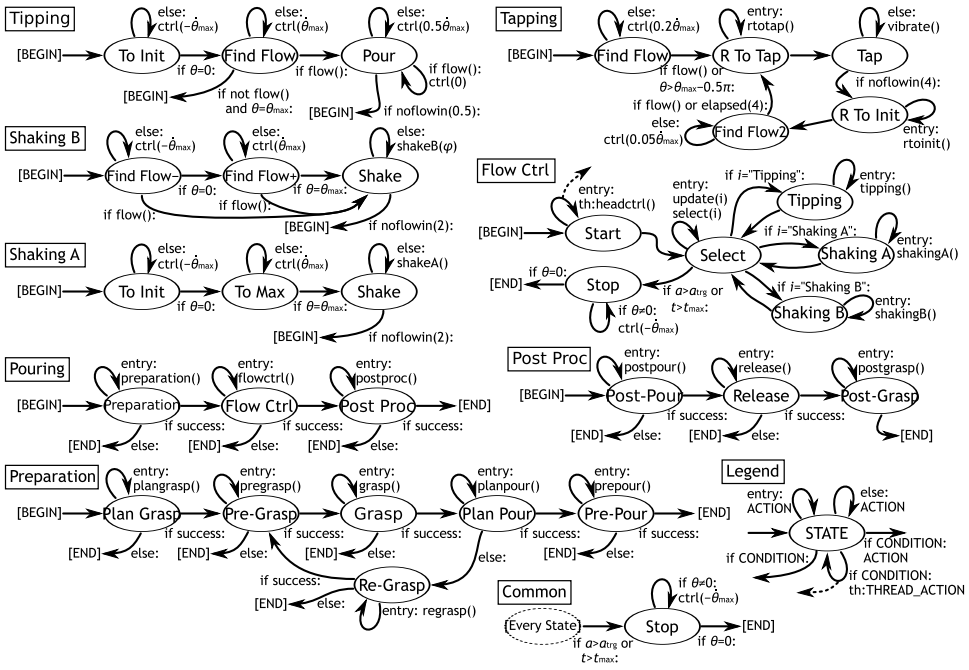


Fig. 7. Finite state machines for pouring. Each state consists of a set of condition and action pairs. There are special notations: `else` denotes a condition that is satisfied if the other conditions are false, `entry` denotes an entry action that is performed when entering the state. `th` denotes a threading execution which realizes parallel state machines. The stop condition state "Common" is used in tipping, shaking A and B, and tapping, which decides to stop if a target amount a_{trg} is achieved, or a maximum duration $t_{\max}$ is exceeded.

A state machine for tipping is illustrated in Fig. 7. We assume $\theta = 0$ at the initial pose, and θ should be less than $\theta_{\max}$ where the source container is upside down. The "To Init" state is used when θ is not zero at the state. This happens when the robot tries again from the initial pose due to jamming, or the pouring skill is combined with other skills. The other states, "Find Flow", "Pour", and "Stop", correspond to Phases 1 to 3 respectively. The utility functions and the constants used in the tipping state machine are: $\texttt{ctrl}(\dot{\theta})$: control θ with its velocity $\dot{\theta}$, $\texttt{flow}()$: true if flow observed, $\texttt{noflowin}(\Delta t)$: true if no flow observed in Δt, and $\dot{\theta}_{\max}$: maximum velocity of θ. Since during grasping, the gripper pose corresponds to the grasp pose, we can compute the edge point in the wrist frame. Using an inverse kinematics solver for the wrist link, we can control θ around the edge point. From the initial pose of the source container, we can estimate the rotation axis.

We expect generalization from this state machine in terms of target amounts, as well as initial amounts in the source, source container shapes, and material types. In the experiments, we investigate these generalization abilities.

5.2.2. *Shaking*

We model the two types of shaking shown in Fig. 4(a) and 4(b) since they have good performance. We refer to them as shaking A and B respectively. Shaking A is a vertical motion while holding the source container upside down. Shaking B is shaking at an angle where the flow is maximized.

Shaking A and B are modeled with finite state machines as illustrated in Fig. 7. In shaking A, θ is increased until the source container becomes upside down ("To Max"). During "To Max", any flow is ignored other than a_{trg} is achieved. Then a shaking motion starts. In shaking B, θ is increased until some flow is observed, then a shaking motion starts. In these state machines, `shakeA`() denotes the shaking A motion where the source is moved in vertical direction, and `shakeB`(ϕ) denotes the shaking B motion where the shaking direction is modified horizontally by an angle parameter ϕ, $[\sin(\phi), 0, -\cos(\phi)]$ defined in the source container's frame. The parameter ϕ is chosen to be suitable for the current situation by a learning method described in Section 6.

5.2.3. *Tapping*

Reproducing a tapping motion with a robot is a bit difficult unless the robot can move a gripper or finger rapidly. The PR2 can "tap" by touching the right gripper to the source container held by the left gripper, and then vibrating the right gripper. Unfortunately the vibration is fairly slow.

The tapping state machine is illustrated in Fig. 7. This state machine is more complicated compared to the others since the tapping includes dual-gripper motions. In preliminary experiments, we found that the initial flow is much larger than the amount poured by tapping. Thus, we design the state machine so that it can control the initial flow accurately. The utility functions used in the tapping state machine are: `rtotap`(): move the right gripper to the tapping pose, `rtoinit`(): move the right gripper to the initial pose, `elapsed`(Δt): true if the elapsed time after entering the state is greater than Δt, `vibrate`(): perform vibrating motion. Note that in the tapping state machine, the "Stop" state in "Common" has an entry action: `rtoinit`().

5.2.4. *General Flow Control*

We model a general flow control where the tipping and shaking A and B skills are unified. The basic idea is defining a higher-level state machine which executes those skills selectively. Selecting a skill for each situation (container shape and material type) is the difficult part of this approach. We use learning and discuss the detail in Section 6. Skill selection and its learning are introduced into the state machine of the general flow control by functions `select`(i) and `update`(i). Once skill i is selected, i is executed in the usual state machine fashion.

The state machine for general flow control ("Flow Ctrl") is illustrated in Fig. 7.

This is a higher-level state machine, where the individual state machines are used as subordinal state machines. Basically, [BEGIN] and [END] of each lower-level state machine are connected to the "Select" state of the flow control state machine. The "Stop" state of each subordinal state machine is removed and the terminal condition is directly connected to "Select", since every subordinal state machine is designed to be able to start at any value of θ.

In addition, we consider the situation of Setup 2 in Section 4.2. When the source container is taller than the receiving container, humans take a different strategy. In such a situation, humans move the mouth of the source container vertically as we described in Setup 2 of Section 4.2. The purpose of this vertical movement may be to avoid collision between the source container and the table, so we can model this movement independently from flow control by using a parallel state machine. In the "Start" state of "Flow Ctrl" in Fig. 7, the controller for the mouth height (`headctrl`()) is executed in parallel, then a skill suitable for the current situation is selected in the "Select" state. Here, `headctrl` is a simple control modeled with a state machine: it moves the source container toward a target height or until the container is too close to the table. Note that this height control also covers the situation where the source container is smaller than the receiving container.

5.3. *Modeling the Complete Pouring Behavior*

In this section we model the entire pouring behavior. We consider its procedural representation, including timing to execute several types of planning schemes, i.e. grasping, pouring locations, collision free paths for arm movement (Setup 1 of Section 4.2), and re-grasping (Setup 3). The details of these planning methods are described in Section 7.

The pouring procedure is represented with a state machine "Pouring", illustrated in Fig. 7. The "Pouring" state machine consists of three steps: a preparation process ("Preparation") where the robot grasps a source container and moves it to the receiving container position, flow control ("Flow Ctrl") mentioned above, and the post process ("Post Proc") where the robot moves the source container to a final destination. Each step is modeled with state machines, also illustrated in Fig. 7. The utility functions used in those state machines are: `plangrasp`(): execute grasp pose planning for the source container, `pregrasp`(): execute a pre-grasping motion which is a state machine to move the gripper from a current pose to the planned grasp pose, `grasp`(): execute a grasping motion to grasp the source container, `planpour`(): execute pouring location planning for the source and a receiving container, `regrasp`(): execute a re-grasping planning and a re-grasping motion, `prepour`(): execute a pre-pouring motion which is a state machine to move the gripper grasping the source container to the pouring location of the receiving container, `flowctrl`(): execute flow control where several skills like pouring by tipping and shaking are used, `postpour`(): execute a post-pouring motion which is the opposite of the pre-pouring motion, `release`(): execute a releasing motion where the grip-

per is opened to release the source container, `postgrasp`(): execute a post-grasping which is the opposite of the pre-grasping motion.

5.4. *Spreading Model*

We consider a spreading variation of pouring including spreading pattern variations mentioned in Setup 4 of Section 4.2.

Spreading is almost the same as the pouring task. In both tasks, the purpose is moving materials from a source container to a receiver. The difference is that in spreading the goal is to cover the receiver surface (e.g. a slice of bread) with the material. We call that two-dimensional trajectory on the surface a *spreading pattern*.

Our procedural modeling of spreading is almost the same as the pouring model shown in Fig. 7. There are two differences: (1) in the pouring location planning step, a spreading pattern is also planned, and (2) the parallel state machine to control the height of the source container (`headctrl`) is augmented by a control to follow the spreading pattern. In spreading, we use only the tipping skill to control the flow of the material. The speed to move the mouth is decided by the flow speed; on each control time step, the desired speed is proportional to the flow speed.

6. Direct Policy Learning

The human demonstrations shown in Fig. 4(a) and 4(b) where the shaking skills are used was a bit difficult for the human. We found several interesting behaviors. First, the human tried some different skills that we are referring to as shaking A and B. Since the human did not know the most suitable skill for the task, the human searched for the best one through trial and error. The human also adjusted some skill-related parameters such as shaking axis and speed.

We think this behavior is a kind of direct policy learning, where some policy parameters are optimized from observed scores (rewards) in an on-line manner. We introduce a parameter optimization architecture into the skill models so that the robot can handle a wider range of pouring tasks.

In the following, we describe the problem setup, optimization methods, and the architecture to introduce the optimization methods into the skill models. There are many other choices to optimize the parameters, for example, using a reinforcement learning method (e.g. [17]). We are focusing on showing an entire solution to the pouring problem, so we chose simple but practical learning methods.

6.1. *Problem Specification*

Though there are several objectives in the pouring task such as pouring reliably as fast as possible and avoiding spills, we focus on pouring speed optimization. The optimized parameters are, for example, skill selection, the shaking axis, and the shaking speed. The problem is to maximize the pouring speed encoded by a score

function with respect to these parameters. We do not have an analytical model between the parameters and the score; the score can be obtained through actual execution using the parameters.

There are two types of parameters to be optimized. One is a discrete parameter that is selected from a set of options (e.g. a set of skills). The other type is continuous parameters that are selected from a single or multi dimensional continuous space (e.g. a shaking axis).

6.2. *Optimization Methods*

Since score functions like the pouring speed are noisy, we need to choose a robust optimization method. For continuous parameter optimization, we use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) proposed by Hansen [18] whose implementation is available on-line[a]. CMA-ES is an evolutionary algorithm that does not require the gradient of the score function.

For the discrete parameter optimization problem, we use Boltzmann selection (also known as the softmax selection) [19] to select an option where each option is evaluated with an upper confidence bound (UCB). Though there are several versions of the UCB, we use the sum of the expected score μ and its standard deviation σ [20]. μ and σ are updated by an exponential moving average scheme. The details can be found in [21].

6.3. *Architecture*

We use these optimization methods in an on-line manner. An on-line parameter optimization consists of three steps: (1) Selecting parameters to be used, (2) Using the parameters and obtaining the score, and (3) Updating based on the score. A natural way to integrate these steps in a state machine is treating them as actions of the state machine.

The optimization method for skill selection is introduced to decide the skill index i in the flow control state machine (Fig. 7). In this case, the selecting and the updating steps are executed as the entry action of the "Start" state. The score is the difference of the amount divided by the execution duration.

Note that the flow control state machine supports trial and error learning during a single pouring trial. For example, if the material jams in the shaking A state machine, it is detected by the `noflowin`(2) condition, and the state moves back to the "Start" of the flow control. Thus, the skill selection can be updated and a new skill selected.

We apply continuous parameter optimization to ϕ of the shaking B behavior. In this case, the selecting and the updating steps are executed right before and right after the `shakeB`(ϕ) action.

[a]`https://www.lri.fr/~hansen/cmaes_inmatlab.html`

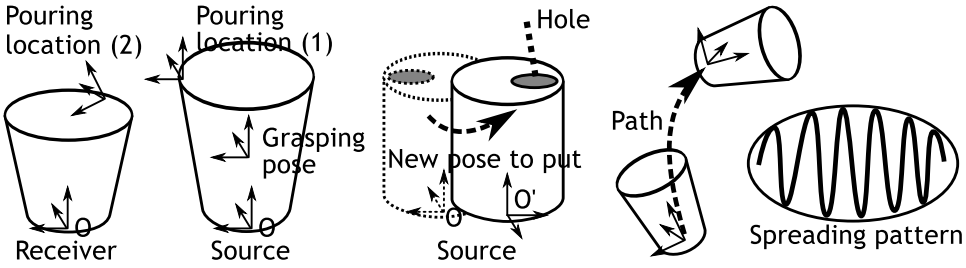


Fig. 8. Planned parameters in pouring.

7. Unified Planning and Learning

In this section, we describe various types of planning in order to realize the demonstrated pouring behaviors of Section 4.2. The parameters to be planned are illustrated in Fig. 8. We also discuss how to improve the planning through learning from practice.

7.1. *Assumptions*

In the following planning methods, we assume that we can use the current robot posture (joint positions), the poses (positions and orientations) of the source and the receiving containers, and the model parameters of the containers. Each container is modeled with a cylinder for a graspable part, a polygon for the mouth, and a bounding box for collision checking. The surface of a receiver for spread materials (e.g. bread) is also modeled as a polygon.

We assume that the robot has an arm with a parallel gripper that has two fingers. The direction of gravity is the negative z-axis in the robot base frame (the torso frame). The robot uses the left arm (if it has two arms) and starts pouring from the left side of the receiving container. We also assume that we can use a collision model of the robot that includes self-collision and collision with objects.

7.2. *Planning using Optimization*

First we explore planning methods based on optimization. Important elements in planning using optimization are (1) parameterization: variables used in planning; and choosing the minimum number of parameters is important for computational efficiency and reducing the number of local minima, e.g. three parameters are enough for grasping a cylinder, and (2) an evaluation function: a function to compute a score for a situation and a parameter vector. Examples of situations are the current robot posture, container poses, and container types. Constraints on parameters (e.g. collision) are included in the evaluation function; the return value is in $\{\mathbb{R}, \phi\}$ where ϕ denotes invalidity of the parameters in that situation.

When these elements are defined, we can apply a general optimization tool.

Here we use CMA-ES [18]. CMA-ES needs an initial guess. Since the evaluation functions may not be feasible due to factors such as a collision, their structures may be complicated. Our learning method mentioned later also makes the evaluation functions more complicated. Thus, a good initial guess is important. Our solution is that we prepare a database where many previous planning results are stored. These points are the parameters and the situations that the robot actually used them. First, we search for parameters from the database that have valid scores. These scores are computed by the evaluation function with the parameters in the database and the current situation. Then, we use the parameters that have the maximum score as the initial parameters of the optimizer. If no valid parameters are found in the database, we search several (e.g. 5) parameter vectors that have valid scores randomly. We choose the best parameters for the initialization of the optimizer. In the actual implementation, we are limiting the number of computations of the evaluation function for speedup. After every successful plan execution, the plan parameters, its score, and the situation are stored in the database.

We apply this planning framework to grasp pose, pouring location, path of moving the arm, and re-grasping parameters.

7.2.1. *Grasp Pose Planning*

Grasp pose planning computes a suitable gripper pose to grasp a container. Let x_{g} denote the grasp pose of the container. Here we also consider a *pre-grasping* pose x_{g0} that is a pose before grasping. The purpose of considering a pre-grasping pose is to simplify the path planning to reach the grasp pose; we can use simpler collision models of containers for path planning to reach the pre-grasping pose, while we need more complicated models for path planning to reach the grasp pose.

Though a grasp pose x_{g} has 6 DoF in general, we use three parameters for a cylinder. Two are orientation parameters, and the last one is a height ratio. The pre-grasping pose x_{g0} has no parameters; its orientation is the same as x_{g}, and the position is behind of x_{g} (the direction of withdrawing the gripper) where the displacement is 1.5 times of the cylinder width.

The evaluation has five aspects. (1) For the asymmetric container case (e.g. a coke can), the center of the mouth is preferred to be on the right side of the gripper (pouring side). (2) The current x, y direction of gripper is preferred to be close to the x, y direction of the grasp pose. (3) The grasp pose is preferred to be parallel to the table. (4) A margin of more than 3 cm is needed between the mouth and the gripper. More than 5 cm is desirable. (5) x_{g} and x_{g0} should be valid. We also check for being collision free and IK solvable. From these aspects, the score is computed.

7.2.2. *Pouring Location Planning*

Pouring location planning computes a pouring location x_{pe} on the source container and a pouring location x_{pl} on the receiving container. These locations include ori-

entation; the x-axis of this local frame is considered as the tipping axis of the flow control. Before flow control starts, x_{pe} is moved to x_{pl}. Similar to grasp pose planning, we also consider a *pre-pouring* location x_{pl0} on the receiving container that is a pose before reaching x_{pl}.

We assume that x_{pe} is on the mouth edge; so its Cartesian position can be parameterized with an angle. Its orientation is automatically decided by considering three assumptions: the x-axis (tipping axis) is along the mouth edge, the z-axis of the orientation is the vertical direction of the polygon plane, and the y-axis corresponds with the vector from the center to the x_{pe} position. x_{pe} is parameterized with a single variable. Similarly, x_{pl} has an angle parameter on the receiver's mouth polygon, but its position is between the point on the mouth edge and the polygon center. This ratio is also a parameter. Additionally, in order to create a margin, its z position is increased 3 cm. The orientation of x_{pl} is decided similarly as x_{pe}, then it is rotated with a small angle around the x-axis. This rotation is to create an initial pouring posture; the angle is typically 45 degrees, but can be modified for each container. There are three parameters in total.

The evaluation has three aspects. (1) The angle between the tipping axis of the receiving container and the x-axis (forward) of the torso should be in the range $[-70, 70]$ degrees, 0 degree is the best. This is to pour from the left side. (2) The angle between the tipping axis of the receiving container and the gripper should be in the range $[-45, 45]$ degrees, 0 degree is the best. (3) When the robot moves x_{pe} to x_{pl0}, the state should be valid.

7.2.3. *Path Planning*

The purpose is to compute a collision-free path from a current gripper pose x_{curr} to a target gripper pose x_{trg}. Though there are many algorithm like RRT [22], here we use a simple method which is adequate for our situation.

For an x, y, z-trajectory, we use a Hermite cubic spline with four knot points that are put on the same plane, and parameterized with five variables. The first knot point is x_{curr} and the last is x_{trg}. The plane that includes x_{curr} and x_{trg} has one DoF, rotation around the line between them; this angle is the first parameter. The remaining parameters are used to decide the middle two knot points, so they are two angles and two normalized distances. For the orientation trajectory, we simply use a linear interpolation of the two orientations[b].

The evaluation considers that all sample points on the trajectory should be valid and a shorter trajectory is better.

[b]Specifically, for two quaternions q_1, q_2, first we compute an axis a and an angle ψ that rotate q_1 to q_2. Then we linearly interpolate the angle from 0 to ψ ($t\psi$, $t \in [0, 1]$) and compute an interpolated orientation by rotating q_1 with a and $t\psi$.

7.2.4. *Re-grasping Planning*

The purpose is to find a new position x_{put} to put the source container when planning the pouring location is hard. This is used to change the hole position of a coke can.

The ideal rotation angle can be computed easily; moving the hole to the right side in the gripper. Let θ_{put} denote this angle. There are three parameters: the displacement of x and y from the current position, and the displacement from the θ_{put}. The evaluation considers the validities of x_{put}, 10 cm above of x_{put}, withdrawing pose from x_{put}, and a roughly estimated new grasp pose. These are used as via points in the re-grasping motion.

7.3. *Other Planning Schemes*

Planning using optimization is a widely applicable approach, but robots need different types of planning to achieve the behaviors that the humans are doing. We have applied an optimization based approach to path planning, however, if a robot needs to plan a path in more complicated situations, this approach will not be efficient since the number of the parameters become large and it is hard to identify the correct number of key points. For those cases, we can use a sampling based planning approach like RRT.

Another path planning example is planning spreading patterns. We use a template pattern and put many templates to cover the whole surface of the receiver. The number of parameters is proportional to the number of templates, so applying the optimization based approach is again inefficient.

Here we describe a simple approach for planning spreading patterns; we compute an x, y-trajectory. We use a template pattern like a single cycle sine curve that has two parameters: the amplitudes of the first and the second peaks. First, we compute the center of the polygon, estimate the direction to repeat the template by applying PCA to the polygon points, and find a start point. The start point is on a line decided by the center and the direction, is inside the polygon, and is far from the center. The amplitude parameters are iteratively optimized by fitting the template so that amplitudes are maximized subject to the constraint that all sample points on the trajectory are inside the polygon.

7.4. *Adding Learning to Planning*

We discuss how to improve planning through learning from practice. In direct policy learning (Section 6), discrete parameters (e.g. selecting a skill) and continuous parameters (e.g. shaking axis) are learned from practice in order to adapt to a new situation. A real number score is given as an evaluation of each parameter. Here we introduce a similar learning architecture into our system. We start with discussing the similarity of planning and learning.

In Section 6, we used a softmax-like method for learning discrete parameters, and CMA-ES [18] for learning continuous parameters. The learning process of the

softmax-like method is: (1) learning a lookup table of scores, (2) selecting a parameter based on the table, and (3) updating the table from actual execution. In CMA-ES, a score function is approximated with a covariance matrix from samples obtained through actual executions. We can regard the lookup table and the covariance matrix as an evaluation function of the parameter. Selecting a parameter in these learning methods is a sort of planning. Updating a score function (evaluation function) is the core of these learning methods, which is missing in our planning with optimization scheme.

Therefore, we introduce a simple method to update an evaluation function through practice. The idea is to use *bad* examples obtained from actual executions in order to modify the evaluation function $f_{\mathrm{eval}}(p, s)$ where the arguments are a parameter p and a situation s. The modification method is similar to the idea of a radial basis function; if the parameter and the situation is close to a bad example, the new evaluation is ϕ (invalid). Thus, the robot will not plan the same parameters in the same situation. We refer to this approach as "poisoning" bad parameter rollouts.

For this method, we need to define a distance function for (p, s) and (p', s'), a threshold, and assessment rules. The assessment rules are used to decide if a planned parameter is bad or not, which is decided according to the effect of the parameter. In our planning models, grasp pose planning has a large effect on the pre-grasping motion and the grasping motion; so if these motions exit in failure, the corresponding grasp pose parameters are assessed as bad. Similarly, pouring location planning has a large effect on the pre-pouring motion, flow control, and the post-pouring motion. Re-grasping planning has a large effect on the re-grasping motion. Re-grasping is assumed to be executed once if necessary, so if the re-grasping is executed twice, those re-grasping parameters are assessed as bad. In addition to these automatic processes, a human operator can assess planned parameters as bad by using manual input in our implementation.

The distance between (p, s) and (p', s') is defined for each planner individually. A common way to compute a distance is as follows: computing the L-2 norms of $p - p'$ and the differences of individual elements in s and s' (let's say, $s_1 - s'_1, s_2 - s'_2, \dots$). Then, we multiply a constant value for each norm for adjustment, and choose the maximum norm as the distance. The elements of a situation are the positions of the source and receiving containers (s_{ps}, s_{pr}), the orientations of the source and receiving containers (s_{os}, s_{or}), initial joint positions (s_q), and target position and orientation (s_{ptrg}, s_{otrg}). The distance definitions of grasp pose, pouring location,

path, and re-grasping planning are respectively:

$$d_{\text{grasp}} = \max(\|p - p'\|, \|s_{ps} - s'_{ps}\|, 0.2\|s_{os} - s'_{os}\|, \|s_{pr} - s'_{pr}\|, 0.2\|s_{or} - s'_{or}\|), \tag{1}$$

$$d_{\text{pour}} = \max(\|p - p'\|, \|s_{ps} - s'_{ps}\|, 0.2\|s_{os} - s'_{os}\|, \|s_{pr} - s'_{pr}\|, 0.2\|s_{or} - s'_{or}\|), \tag{2}$$

$$d_{\text{path}} = \max(\|p - p'\|, \|s_{ptrg} - s'_{ptrg}\|, \|s_{otrg} - s'_{otrg}\|, \|s_q - s'_q\|_\infty), \tag{3}$$

$$d_{\text{regrasp}} = \max(10\|p_{xy} - p'_{xy}\|, |p_\theta - p'_\theta|, \|s_{ps} - s'_{ps}\|, 0.2\|s_{os} - s'_{os}\|). \tag{4}$$

We are using the maximum norm between joint position vectors. In the distance of re-grasping planning, since the x and y displacement p_{xy} should be more sensitive than the angle displacement p_θ, they are separated and have different weights.

This distance measure computes a very situation-specific value. We can not expect wide generalization of the modification. The reason we chose this approach is that we already have an evaluation function for each planning method, so the improvement from actual practice is not that much. Therefore, instead of using a distance function with a wider generalization ability, that has a possibility of undesired generalization, we chose a situation-specific one.

8. Experiments

We implement the pouring skills modeled in the previous sections on a robot, and investigate their capabilities. We use a PR2 robot that has two 7-degrees of freedom arms with grippers. We use ROS packages[c] for the PR2 to implement low-level control, collision detection, and inverse kinematics solver of the grippers. The accompanying video is available at `http://youtu.be/GjwfbOur3CQ`.

8.1. *Flow Control*

First, we show the generalization ability of flow control with tipping in terms of target amounts. Second, we compare shaking A and B skills. Third, we investigate the tapping skill. Fourth, we show how learning from practice works. Finally, we demonstrate the generalization ability of the general flow control model in terms of source container shapes and material types. Fig. 9(a) shows the setup of the robot and containers.

In order to measure the amount of poured material, we use an RGB camera and detect specific colors as shown in Fig. 9(b). The ratio of colored areas is used as the amount. For this purpose, as the receiving container, we use a transparent plastic container whose back half is colored.

We prepare 14 containers and 5 materials as shown in Fig. 10. Though we use only dry materials to avoid hardware damage by liquids, they behave similarly to various types of liquids.

In the following experiments, we focus on flow control; the robot starts to pour when the robot is grasping a source container and holding it near the receiver.

[c] `http://ros.org/`

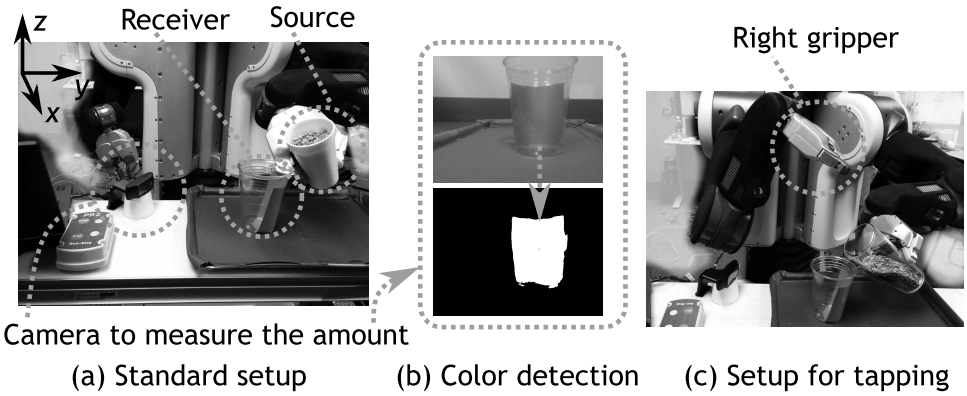


Fig. 9. Setup of the experiments.

Fig. 10. Source containers and poured materials. Each photo contains a coin of diameter 0.955 inches as a scale. BBs are copper-coated bullets for toy guns. The container B3 and B4 are specially designed for the experiments; these holes are designed to be small in order to produce jamming.

8.1.1. *Tipping*

Fig. 11 shows a typical result of pouring where the target amount is 0.5, the source container is B1, and the poured material is the dried peas. The flow started around 2 [s], and the robot slows down. Compared to the human demonstration

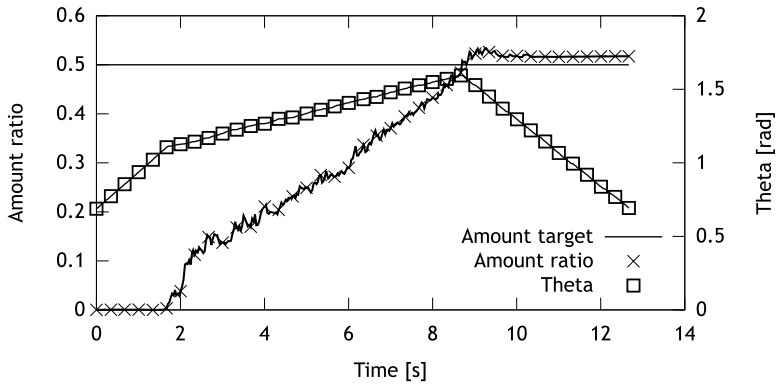


Fig. 11. Typical result of pouring. The setup corresponds to that of the human demonstration (Fig. 3); the source container is B1, and the poured material is the dried peas. For the consistency with Fig. 3, θ (Theta) is shifted so that their ranges match with each other.

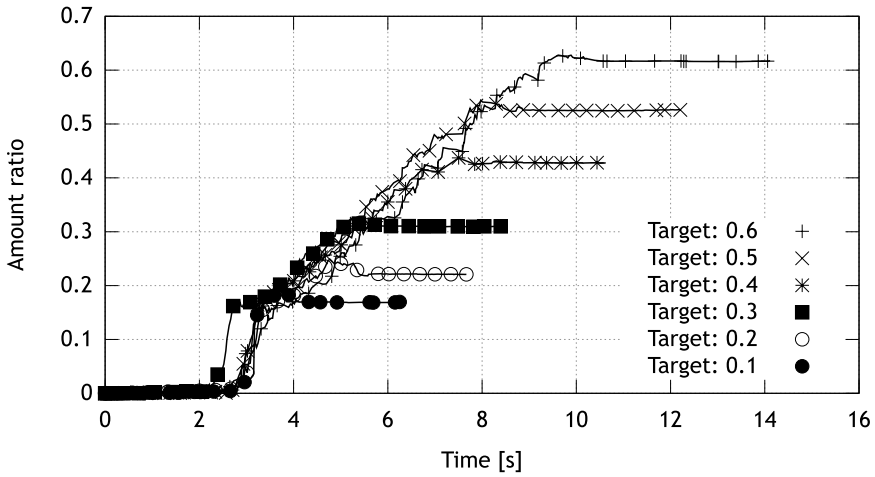
(Fig. 3), the robot behavior has a similar structure; namely, there are three phases. However, we can find some differences. For example, the human slows down the angular velocity before the flow starts. One possible reason is that the human tries to keep the initial flow small. Estimating the orientation where the flow starts is necessary to reproduce this behavior; humans are using visual information and/or force information. We have not yet implemented this behavior.

Next, we investigate the generalization ability of the pouring skill in terms of target amounts. We use B1 as the source container, the dry peas and the rice as the poured material, and change the target amount from 0.1 to 0.6. Fig. 12(a) and 12(b) show the results of using the peas and the rice respectively. In Fig. 12(a), the case of the largest error is at the target of 0.1. The reason is that the amount of initial flow (around 3 [s]) was large; the initial flow poured more than the target amount. In the other cases, the target amount is achieved more closely.

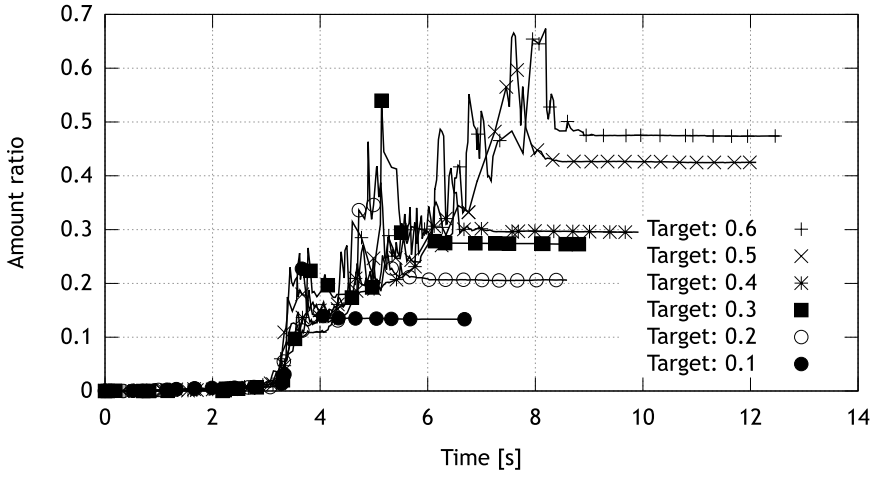
On the other hand, the results of the rice case (Fig. 12(b)) seem to be noisy. Each amount trajectory overshoots. This was caused by the vision system being confused by the stream of material during the pouring. Compared to the peas, the flow of rice spread more widely and was more visible to the camera. There are several ways to reduce this problem: using a better vision system, adjusting pouring parameters, using force information [11], or using other sensors. We will improve our sensing in future work. However, in both cases the pouring skill has some generalization ability in terms of target amounts.

8.1.2. *Comparison of Shaking Skills*

We investigate two versions of the shaking skills, shaking A and B, and make clear the necessity of both versions. We use B4 and B3 as the source containers. The poured material is the dried peas. For each container, we apply shaking A and



(a) Dry peas.



(b) Rice.

Fig. 12. Generalization in terms of target amounts.

shaking B with axes $\phi = 0$ and $\phi = \pi/4$.

Table 2 shows the results of the B4 and the B3 cases. The table describes the number of failures out of 3 runs, and average pouring duration. The failure is a timeout case; $t_{\max}$ is about 40 [s] in the B4 case, and is about 80 [s] in the B3 case, since the latter case is more difficult. Obviously, in the B4 case, shaking A outperforms the others; meanwhile in the B3 case, shaking B with $\phi = \pi/4$ is the best. Both shaking versions are necessary to cover a wide range of source containers.

Fig. 13(a) shows a result of shaking A for the B4 container. Fig. 13(b) shows a

Table 2. Comparison of shaking A and shaking B.

Src.	Method	# of failures	Avg. duration [s]
B4	Shaking A	0	19.06
	Shaking B(0)	2	45.76
	Shaking B($\pi/4$)	0	31.03
B3	Shaking A	2	94.29
	Shaking B(0)	3	N/A
	Shaking B($\pi/4$)	0	38.87

result of shaking B with $\phi = \pi/4$ for the B3 container. In Fig. 13(a), we can see a little flow before starting shaking (around 5 [s]) but the flow stops due to the jammed material, so tipping does not work any more. During shaking, we can see the amount is increasing. Thus, shaking is a possible way for the robot to solve jamming. Compared to the B4 case, it takes more time to pour the target amount in the B3 case. The B3 container is also difficult for humans to pour.

8.1.3. *Tapping*

Next, we investigate the performance of tapping. This skill uses the right gripper to tap, so we start from the setup shown in Fig. 9(c). The robot grasps a source container with the left gripper while the right gripper stays above the receiving container. In order to touch the right gripper to the source container, we define a tapping pose as a constant vector in the source container frame. We use the B25 container and the BBs.

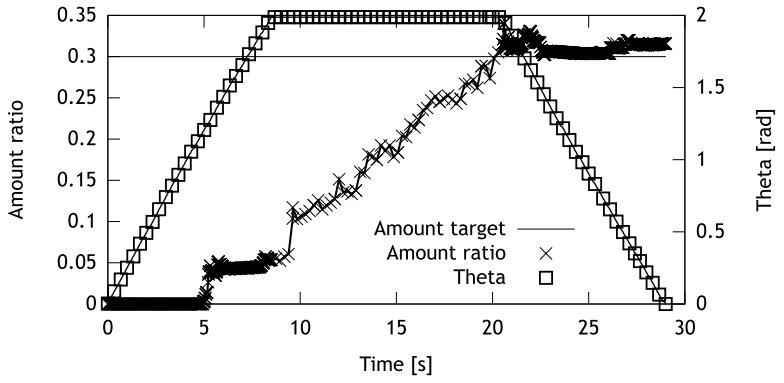
Fig. 14 shows the result of tapping where the result of B25-peas in the previous tipping experiment is shown as the comparison. We can see that using tapping, the robot can pour the material very slowly. Thus, tapping enables the robot to pour accurately. However, there is another difficulty; in the slow pouring setup, the amount of initial flow is comparably large, which dominates the total amount. Thus, without an accurate controller for the initial flow, we cannot achieve accurate pouring with respect to the total amount.

8.1.4. *Direct Policy Learning in Flow Control*

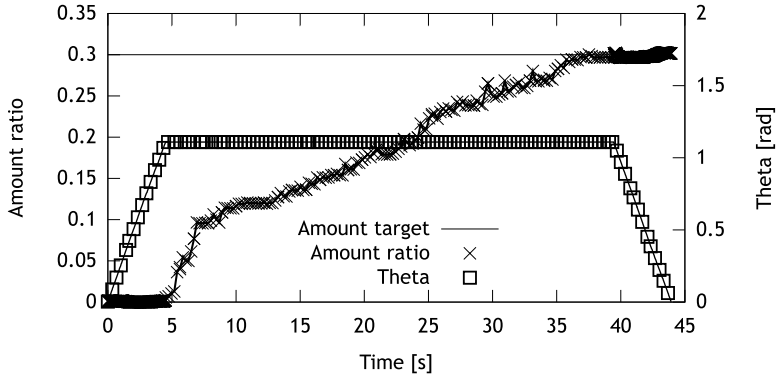
We demonstrate how the parameter optimization architecture works. Here, we investigate separately discrete parameter optimization (skill selection) and continuous parameter optimization.

Skill Selection Optimization In this experiment there are three choices: tipping, shaking A, and shaking B. The axis of shaking B is fixed to $\phi = \pi/4$. We initialize the expected scores of the three options as 1.0, 0.5, 0.5 respectively, which means that the robot will use tipping initially.

Fig. 15(a) shows the result of the B3 case, and Fig. 15(b) shows the result of the



(a) Shaking A. The source container is B4.



(b) Shaking B(π/4). The source container is B3.

Fig. 13. Results of shaking A and B. The material is the dried peas. While the orientation θ takes a constant value, the shaking motion is performed.

B4 case. The dried peas are used in both cases. Several trials are sequentially done in each case; six trials in the B3 case, and seven trials in the B4 case. In each graph, the selected option is plotted on the amount trajectory.

In the first trial of Fig. 15(a), we can see the three options were tried. First, the robot applied tipping, but since it did not work, the robot switched to shaking A (recall that this is an on-line parameter optimization). In the first and second trials, shaking A seems to have been dominant. However, shaking A got stuck due to jamming in the 4-th trial. Eventually, shaking B was selected.

In the first trial of Fig. 15(b), only tipping was used though the selection was done several times. As we could see a little flow before starting shaking in Fig. 13(a), tipping works in this setup initially. Actually in the first trial, alternately tipping and going back to the initial position achieved the target amount. Thus, it took several times to learn that the performance of tipping was not good. In the second

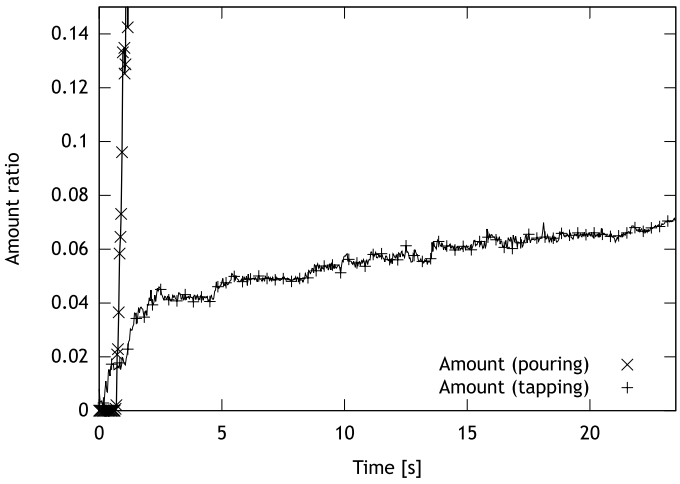


Fig. 14. Comparison of tipping and tapping.

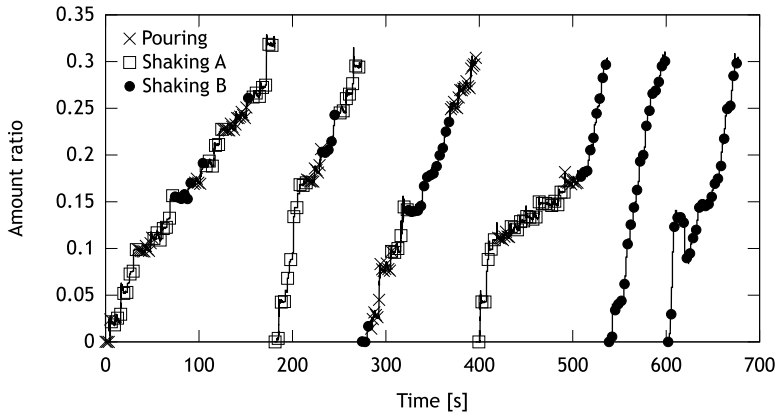
and the third trials, the robot used shaking B and A respectively. Since the robot could pour continuously with each skill, it did not change the skill in each trial. In 5-th and 6-th trials, the robot experienced jamming with the shaking B. Eventually, the robot decided to use the shaking A for this situation.

Therefore, in both the B3 and B4 cases, we obtain the corresponding results with the previous experiment.

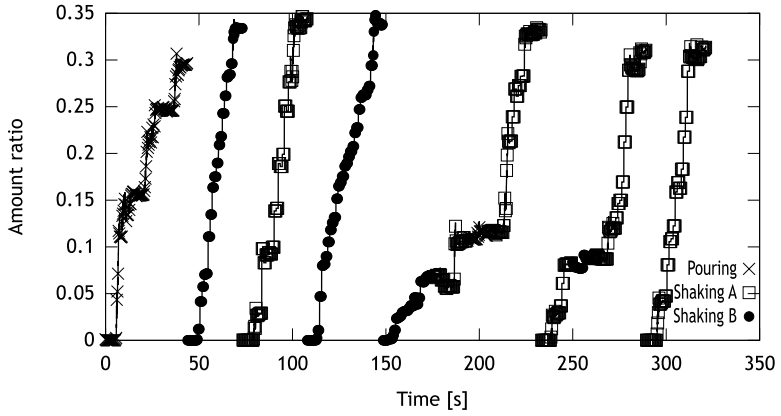
Continuous Parameter Optimization Next, we optimize the parameter ϕ to decide the axis of shaking B. The initial mean and standard deviation are $\phi = \pi/4$ and 1 respectively. In the early stage of optimization, the robot will choose the parameter almost randomly with this configuration. ϕ is limited in $[0, \phi/2]$.

We ran 11 trials sequentially. Fig. 16(a) shows the scores in each generation. Fig. 16(b) shows the parameters in each generation of the CMA-ES; there are 4 individuals (different parameters) in each generation. We can see that the parameter converges to around 0.6 in Fig. 16(b), and the score is improved in Fig. 16(a). The pouring duration was improved from 60.71 [s] of the first trial to 42.47 [s] of the last trial. In the previous experiment, we compared $\phi = 0$ and $\phi = \pi/4$ in the same setup, and found that $\phi = \pi/4$ is better. In this experiment, the robot could find an optimal parameter automatically, and the resulting parameter is close to the manual optimization result, $\phi = \pi/4$. Thus, the CMA-ES could find an appropriate solution.

On the other hand, the score function was very noisy. Even taking the same parameter, the resulting scores are different. The shaking result is affected by the previous shaking motion. Due to these effects, in Fig. 16(b), the parameter seems to have almost converged around 30 [s], but the score of the corresponding generation



(a) Container B3.



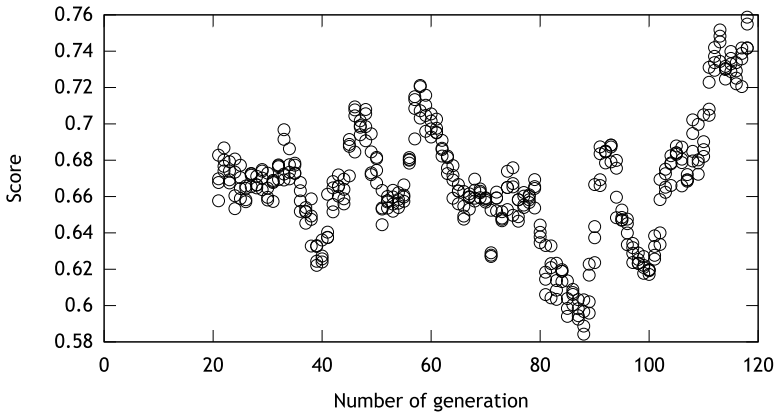
(b) Container B4.

Fig. 15. Learning process of skill selection optimization.

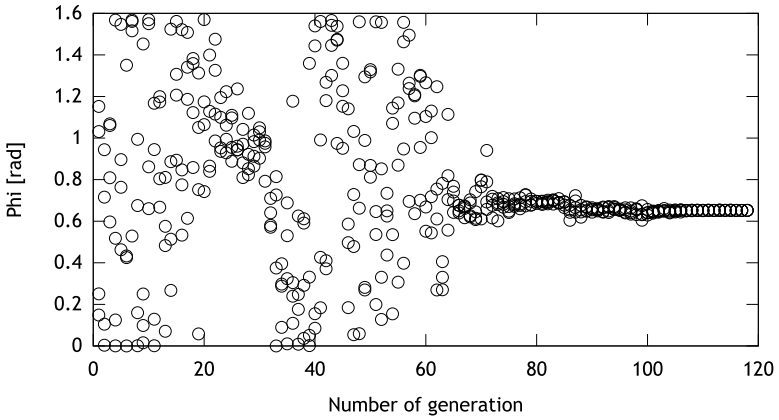
was not so high. After that, it increased the search deviation and found a parameter with a better score. Thus, CMA-ES is practically useful.

8.1.5. *Generalization of the Flow Controller*

We investigate the generalization ability in terms of source container shapes and material types. Each situation is described as a SOURCE-MATERIAL format; e.g. B1-BBS. We prepared 15 combinations from the containers and the materials shown in Fig. 10. For some of them, we manually assigned the parameters of the skill selection and the shaking axis. We assigned incorrect parameters for a B3-rice case where we used the same parameters as those of B3-peas treated in the previous experiments. We assume the parameters of the other cases are unknown. For these



(a) Scores per generation (larger is better). A moving average filter is applied, and the data of the first 20 generations is omitted because of inadequate data for the filter.

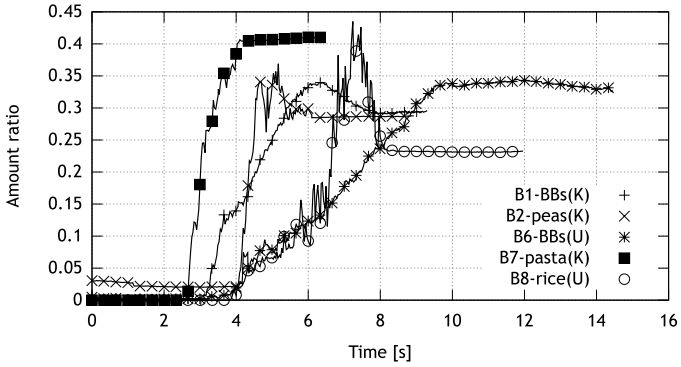


(b) Chosen parameters (ϕ) per generation.

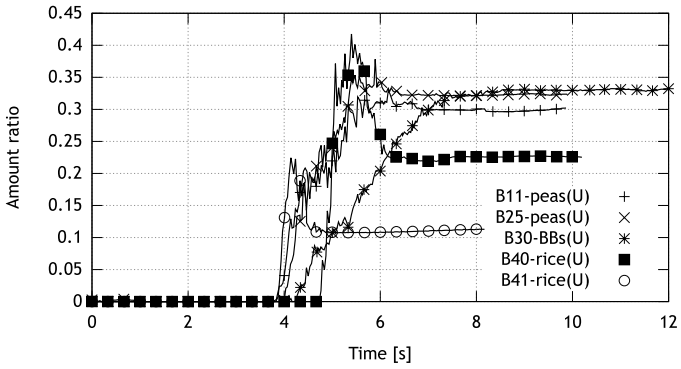
Fig. 16. Learning process of parameter ϕ optimization in shaking B.

cases, we initialized the parameters as was done in the parameter optimization experiments. The target amount is 0.3 except for a B41-rice case; in the B41-rice case, the target amount is 0.1 since B41 is a small container.

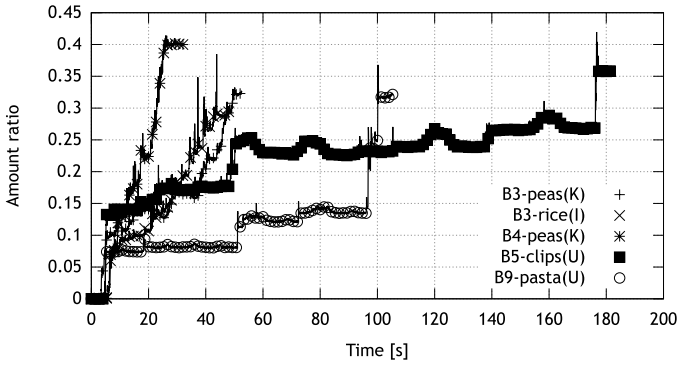
Fig. 17 shows the results where the combinations are categorized into long pouring duration ones (Fig. 17(c)) and short duration ones (Fig. 17(a) and Fig. 17(b)). In the B8-rice, the B40-rice, and the B41-rice cases, the overshoot problem happened similarly to the previous experiment. In the B7-pasta case, the poured amount exceeds the target significantly. This is because the friction between the source container and the material is pretty low in this combination. In the B6-BBs case, the poured amount also exceeds the target. The reason is that since the hole of the



(a) Short pouring duration (1).



(b) Short pouring duration (2).



(c) Long pouring duration.

Fig. 17. Results of generalization test in terms of source container shapes and material types. Each curve shows a result of SOURCE-MATERIAL combination. (K) indicates the parameters are known, (U) indicates the parameters are unknown, and (I) indicates the original parameters are incorrect.

Fig. 18. Containers and their models. Each model has a graspable cylinder, a polygon of the mouth, and the bounding box. The mouths of the containers B54, B56, and B58 are not located at the center.

container B6 is small, the robot rotated the container more than the others, which increased the material flow during moving the container back to the initial orientation. For the other cases in the short pouring duration category, the target amount was almost achieved.

In the long pouring duration cases in Fig. 17(c), the known cases, B3-peas and B4-peas, were poured relatively quickly. Though incorrect parameters were given, the B3-rice case was also poured quickly. This is because the problem setup was similar; the difference was the dried peas and the rice only. The B5-clips and the B9-pasta cases took longer. These problems were difficult since the particle was large compared to the containers' holes. Since these were unknown setups, the skill selection was optimized during the execution (we executed only one trial).

Though there is room for improvement, we have achieved some generalization.

8.2. *Planning and Learning in the Entire Pouring Process*

Next, we discuss the simulation experiments to investigate whether the state machines and the planning methods work as expected. Then we discuss the robot experiments.

Fig. 18 shows the containers and their models used in the experiments. The container models were measured manually. The measured parameters were the Cartesian coordinates of the end points of the graspable cylinder and its width, and the points of the polygon of the mouth which is assumed to be a circle or ellipse segment. These are defined in the container's local frame. The bounding box of a container is automatically computed. Fig. 18 also shows the materials used in the experiments.

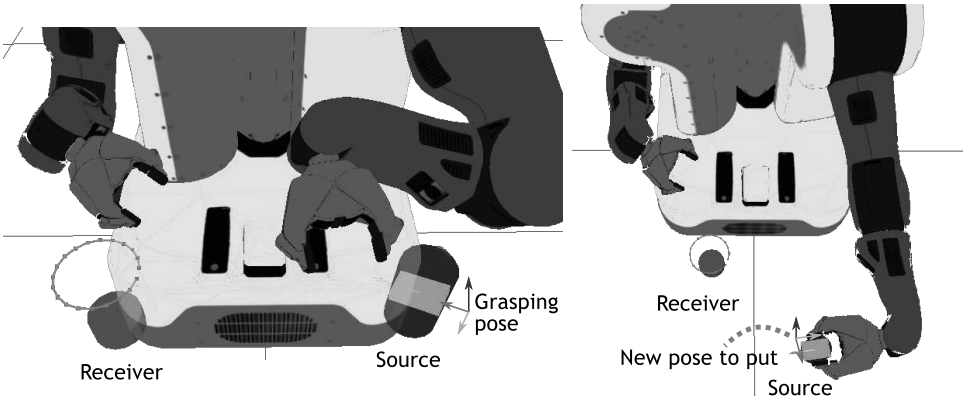


Fig. 19. Failure cases of pouring in simulation. The left is the failure in the pre-grasping motion where the robot needs to move the left gripper to the grasp pose. The right is the failure in the re-grasping where the robot needs to put the grasping container at the new pose. The source containers are B54 and B58 respectively; like a coke can, both have mouth holes whose locations are not the center, so the planned grasp poses seem to be cramped.

8.2.1. *Simulation*

We use the Gazebo simulator[d] to simulate the robot motions. Here, we apply the dynamics simulation only for robot control. We treat the other objects, i.e. a table and containers, as virtual objects. Thus, we can compute the collisions among these objects and the robot, but the collisions do not have a physical effect.

We put a table in front of the robot, and put a receiving container and a source container at random poses on the table. We use ten types of source containers, and execute the pouring state machine 10 times for each source container.

Table 3 shows the simulation results. The average duration of staying in each state, the number of executions, and the number of failures are shown. We did not simulate flow control, so its results are not available. Our pouring model seems to be working correctly at this level, but there were several failures. The reason for failures in the grasp pose planning was that the initial pose of a source container was infeasible, so the robot could not find a solution.

The other failures in the pre-grasping motion, the pre-pouring motion, the post-grasping motion, and the re-grasping were failures of path planning; the conditions were too difficult to plan the path. Using a different planner that has more capability will provide a solution to these conditions, but we think the problem is the planning of the other parameters. Fig. 19 shows the conditions of some failure cases. In both cases, it seems to be better to improve the grasp pose planning.

Our learning mechanism mentioned in Section 7.4 solves this problem; the robot can memorize this situation and the parameters as a bad example, which changes

[d]`http://gazebosim.org/`

Table 3. Simulation results.

State	Avr. duration	# of executions (failures)
Grasping pose planning	9.13	146 (4)
Pre-grasping motion	13.43	142 (1)
Grasping motion	1.78	141 (0)
Pouring location planning	4.47	141 (0)
Pre-pouring motion	27.47	94 (1)
Flow control	N/A	93 (0)
Post-pouring motion	14.27	93 (0)
Releasing motion	1.11	93 (0)
Post-grasping	12.04	93 (1)
Re-grasping	81.19	47 (1)
Total	134.22	100 (8)

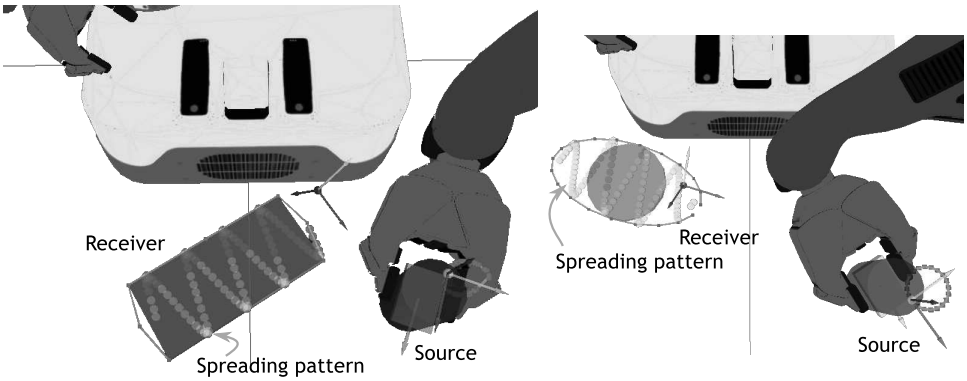


Fig. 20. Planning result of spreading pattern (in simulation).

the planning. In the failed re-grasping case, the human operator taught the robot that the planned grasp pose was bad, since the robot cannot distinguish which of the grasp pose planning and the re-grasping planning was bad. After teaching several bad examples, the robot could succeed to pour in these setups.

Next, we investigate the spreading model in simulation. Since the difference between pouring and spreading is the movement of the source container's mouth, we show the planning result of spreading patterns. Fig. 20 shows examples of planned spreading patterns. The robot could plan spreading patterns for each receiver's shape.

8.2.2. *Robot Experiments*

We investigate our pouring and spreading models with a PR2 robot. Here, we conduct the experiments involving the whole procedure.

Fig. 21 shows the setup of the experiments. We use an external RGB camera to

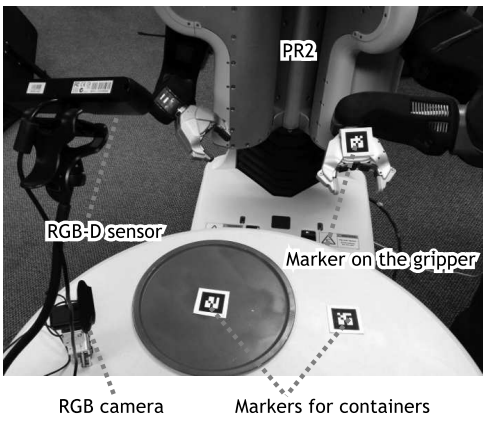


Fig. 21. Setup of the robot experiments.

measure the material amount and an external RGB-D sensor, Xtion Pro Live, to measure the relative poses of containers from the robot.

As in the previous experiments, we use the color information from the RGB camera to estimate the amount of the materials poured into the receiving container. In addition to this, we also measure the flow speed for the spreading control to decide the speed of moving the source container's mouth. We apply an optical flow detection algorithm, the Lucas-Kanade method [23] implemented in OpenCV[e], to the material flow; the measured value of the optical flow has a different scale from the amount measurement, so we apply a simple scaling to the optical flow.

Before each experiment, we put two AR-markers on the table and measure their poses using a ROS package[f], then put a source and a receiving containers on them. Another marker is attached on the robot's left gripper, which is used to calibrate the sensor position in the robot's frame. Since the transformation between the sensor and the robot is not constant[g], that calibration is applied in every time step with a small update ratio. Especially in the pre-grasping motion, the robot stops in front of the source container to wait until the calibration error becomes less than a threshold, then moves for grasping.

We conducted the experiments using B53-peas, B54-BBs, B55-peas, B56-peas, B57-rice, B58-peas, B59-rice, B60-BBs, B61-rice, and B62-peas as source containers and materials. For each setup, we tested two types of initial poses; the source container is left or right. We did not use fixed poses, but each pose was moved a little bit manually.

Fig. 22 shows the snapshots of pouring using the B53 source container. We can see that the whole procedure was completed. During flow control, we can see not only

[e] `http://opencv.org/`
[f] `http://wiki.ros.org/ar_track_alvar`
[g] A possible reason is the lens distortion of the sensor.

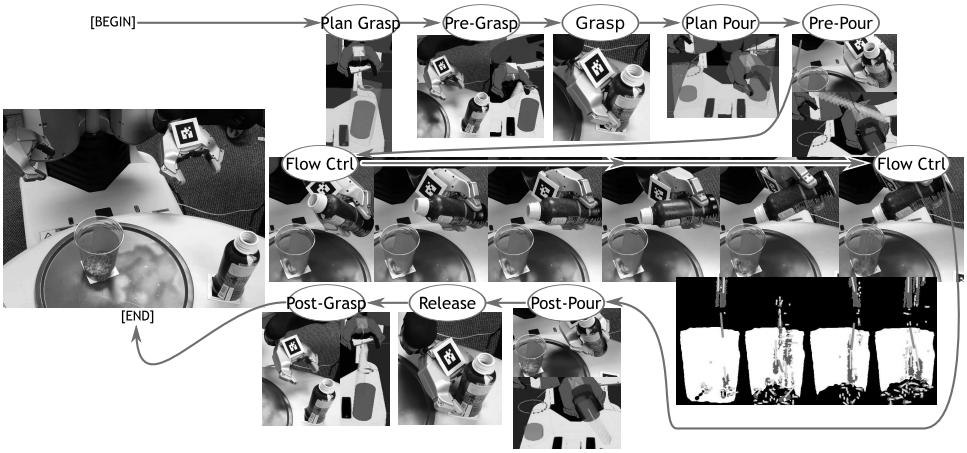


Fig. 22. Snapshots of pouring peas from the B53 source container. Snapshots are put with the corresponding state transitions. In states where the planning is executed, the model view is shown. The snapshots of the RGB camera are also shown in flow control, where the white part indicates the detected colors of the receiving container, and the vertical small lines show the flow.

Table 4. Results of real robot experiments.

State	Avr. duration	# of executions (failures)
Grasp pose planning	12.31	21 (2)
Pre-grasping motion	19.09	19 (3)
Grasping motion	1.68	16 (0)
Pouring location planning	3.06	16 (0)
Pre-pouring motion	17.83	15 (1)
Flow control	77.82	14 (0)
Post-pouring motion	15.70	14 (0)
Releasing motion	0.58	14 (0)
Post-grasping	10.46	14 (0)
Re-grasping	37.56	1 (0)
Total	133.07	20 (6)

tipping the container, but also moving the height of the pouring location (mouth of the container), which was controlled by the parallel state machine.

Table 4 shows the results where the average duration of staying in each state, the number of executions, and the number of failures are shown. In terms of the average duration, there are not big differences from the simulation results shown in Table 3. There were six failures in total, which can be categorized into three types: (1) Pose estimation error of the RGB-D sensor and the AR marker. Especially in the B56-left and B60-right cases, the calibration mentioned above failed during the pre-grasping motion. In these cases, the robot could not move the gripper to the planned grasp pose. In B57-left, the initial pose estimation with the marker of

the source container was inaccurate, which caused the failure in planning the grasp pose in B57-left.
(2) Collision model error. In the B62-right case, this caused the failure of the grasp pose planning.
(3) Planning problem. In B56-right and B58-right cases, the pre-grasp pose x_{g0} obtained in the grasp planning was detected as IK-unsolvable in the succeeding path planner. The grasp planner considers the IK-solvability of x_{g0}, so we did not see this problem in the simulation experiments. However, if the optimizer returns a solution near the boundary of IK solvable and unsolvable, that solution would be affected by sensor noise. We believe this happened in the B56-right and B58-right cases.

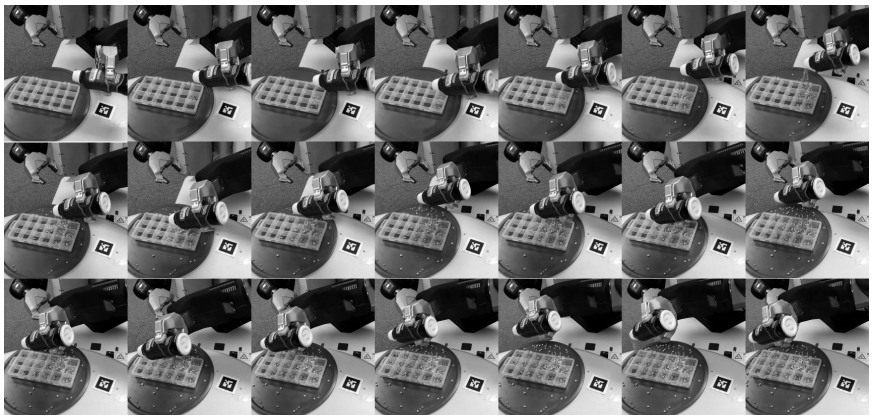
The problem (3) can be solved by using our learning scheme. When the pre-grasping motion fails because of the path planning failure, the previous grasping planning is assessed as bad. The robot will avoid the same solution to the same situation. We investigated this by using the B56-right case. We prepared the same setup as that case, then executed pouring several times. In the second time, the robot output a feasible pre-grasping pose.

We conclude that our pouring behavior can generalize in terms of container shapes, material types, and initial poses of the containers.

Next, we conduct experiments to see how the spreading model works. We use the B53 container as a source container, and the B100 container and a round plate as receivers. Fig. 23 shows the spreading control for each container case. The other procedures are performed as those of pouring. We can see that for different shapes, the spreading patterns are fitted. However, the density of each area is not constant; when spreading tomato sauce on pizza bread, we expect that the density will be constant. In order to solve this issue, we need to improve the accuracy of flow speed control and the control of following the desired spreading pattern.

9. Conclusion

We created a behavior for a general pouring task with several variations: targets, material types, container shapes, initial poses of containers, and target amounts. A major challenge to achieve these variations was flow control since modeling the dynamics of material flow is difficult. We solved this problem by using a skill library where different behaviors for flow control, such as tipping, shaking, and tapping, were stored. Selecting an appropriate behavior for each situation was realized by learning. Planning methods were introduced that handled the variations in targets, container shapes, and initial container poses. In order to increase the planning performance, learning-from-practice methods were also introduced, where the evaluation functions for planning were updated using samples obtained through execution. The simulation and robot experiments using a PR2 robot demonstrated that our pouring behavior could generalize across a variety of pouring variations. This work provides guidance for developing better algorithms and theories about how to



(a) B100.

(b) Circle plate.

Fig. 23. Snapshots of spreading.

perform complex tasks with many variations that go beyond manipulation of rigid objects.

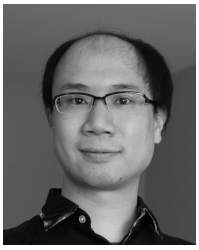
Acknowledgements

We would like to appreciate Dr. Scott Niekum in Carnegie Mellon University who assisted our learning-from-demonstration research. We are also thankful to Professor Maxim Likhachev's Search-based Planning Lab in Carnegie Mellon University for making their PR2 robot available for experiments. A. Yamaguchi was funded in part by Japan Society for the Promotion of Science under the Strategic Young Researcher Overseas Visits Program for Accelerating Brain Circulation G2503.

References

1. P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, 2010, pp. 3232–3237.
2. J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *the IEEE International Conference on Robotics and Automation (ICRA'10)*, 2010, pp. 2308–2315.
3. M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus, "Interpreting and executing recipes with a cooking robot," in *the 13th International Symposium on Experimental Robotics*, 2013, pp. 481–495.
4. M. Phillips, B. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs," in *Robotics: Science and Systems (RSS'12)*, 2012.
5. A. Billard and D. Grollman, "Robot learning by demonstration," *Scholarpedia*, vol. 8, no. 12, p. 3824, 2013.
6. M. Mühlig, M. Gienger, S. Hellbach, J. J. Steil, and C. Goerick, "Task-level imitation learning using variance-based movement optimization," in *the IEEE International Conference on Robotics and Automation (ICRA'09)*, 2009, pp. 1177–1184.
7. P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *the IEEE International Conference on Robotics and Automation (ICRA'09)*, 2009, pp. 763–768.
8. M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011.
9. K. Kronander and A. Billard, "Online learning of varying stiffness through physical human-robot interaction," in *the IEEE International Conference on Robotics and Automation (ICRA'12)*, 2012, pp. 1842–1849.
10. O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A kernel-based approach to direct action perception," in *the IEEE International Conference on Robotics and Automation (ICRA'12)*, 2012, pp. 2605–2610.
11. L. Rozo, P. Jiménez, and C. Torras, "Force-based robot learning of pouring skills using parametric hidden Markov models," in *the IEEE-RAS International Workshop on Robot Motion and Control (RoMoCo)*, 2013.
12. S. Brandi, O. Kroemer, and J. Peters, "Generalizing pouring actions between objects using warped parameters," in *the 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids'14)*, Madrid, 2014, pp. 616–621.
13. D. C. Bentivegna, "Learning from observation using primitives," Ph.D. dissertation, Georgia Institute of Technology, 2004.
14. B. da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," in *the 29th International Conference on Machine Learning (ICML'12)*, 2012.
15. L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.
16. M. Zucker and J. A. Bagnell, "Reinforcement planning: RL for optimal planners," in *the IEEE International Conference on Robotics and Automation (ICRA'12)*, 2012, pp. 1050–4729.
17. J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, pp. 361–379, 2012, 10.1007/s10514-012-9290-3.
18. N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new*

evolutionary computation, J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Springer, 2006, vol. 192, pp. 75–102.
19. R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
20. O. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.
21. A. Yamaguchi, J. Takamatsu, and T. Ogasawara, "Learning strategy fusion to acquire dynamic motion," in *the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids'11)*, Bled, Slovenia, 2011, pp. 247–254.
22. S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. TR 98-11, 1998.
23. B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *the 7th international joint conference on Artificial intelligence (IJCAI'81)*, 1981, pp. 674–679.



Akihiko Yamaguchi received the BE degree from the Kyoto University, Kyoto, Japan, in 2006, and the ME and the PhD degrees from Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2008 and 2011, respectively. From April 2010 to July in 2011, he was with NAIST as a JSPS, Japan Society for the Promotion of Science, Research Fellow. From August 2011 to March 2015, he was with NAIST as an Assistant Professor of the Robotics Laboratory in the Graduate School of Information Science. From April 2014 to March 2015, he was a visiting scholar of Robotics Institute in Carnegie Mellon University, and from April 2015 to present, he is a postdoctoral fellow of the same institute. His research interests include motion learning of robots, reinforcement learning application to robots, machine learning, and artificial intelligence.

Christopher G. Atkeson is a Professor in the Robotics Institute and Human-Computer Interaction Institute at Carnegie Mellon University. He received the M.S. degree in Applied Mathematics (Computer Science) from Harvard University and the Ph.D. degree in Brain and Cognitive Science from M.I.T. He joined the M.I.T. faculty in 1986, moved to the Georgia Institute of Technology College of Computing in 1994, and moved to CMU in 2000.

Tsukasa Ogasawara received the BE, ME and PhD degrees from the University of Tokyo, Tokyo, Japan, in 1978, 1980 and 1983, respectively. From 1983 to 1998, he was with the Electorotechnical Laboratory, Ministry of International Trade and Industry, Ibaraki, Japan. From 1993 to 1994, he was with the Institute for Real-Time Computer Systems and Robotics, University of Karlsruhe, Karlsruhe, Germany, as a Humboldt Research Fellow. In 1998, he joined Nara Institute of Science and Technology, Nara, Japan, as a Professor of the Robotics Laboratory in the Graduate School of Information Science. He is a dean of the Graduate School of Information Science since 2013. His research interests include human-robot interaction, dexterous manipulation and biologically inspired robotics.