THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2014-03

Towards Imitation Learning of Dynamic Manipulation Tasks: A Framework to Learn from Failures

Joshua D. Langsfeld, Krishnanand N. Kaipa, Rodolphe J. Gentili, James A. Reggia, Satyandra K. Gupta

The Institute for Systems Research

A. JAMES CLARK SCHOOL OF ENGINEERING

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the A. James Clark School of Engineering. It is a graduated National Science Foundation Engineering Research Center.

www.isr.umd.edu

Towards Imitation Learning of Dynamic Manipulation Tasks: A Framework to Learn from Failures

Joshua D. Langsfeld, Krishnanand N. Kaipa, Rodolphe J. Gentili,
James A. Reggia, Satyandra K. Gupta[1]

Technical Report, Simulation-Based System Design Laboratory,
University of Maryland, College Park, MD 20742, June 2014.

[1]Corresponding Author

Langsfeld, J.D., Kaipa, K.N., Gentili, R.J., Reggia, J.A., and Gupta, S.K., "Towards imitation learning of dynamic manipulation tasks: A framework to learn from failures," Technical Report, Simulation-Based System Design Laboratory, University of Maryland, College Park, June 2014.

Abstract

We present an imitation learning approach for a dynamic fluid pouring task. Our approach allows learning from errors made by humans and how they recovered from these errors subsequently. We collect both successful and failed human demonstrations of the task. Our algorithm combines a support vector machine based classifier and iterative search to generate initial task parameters for the robot. Next, a refinement algorithm, capturing how demonstrators change parameters to transition from failure to success, enables the robot to address failures. Experimental results with a physical robot are reported to illustrate our approach.

1 Introduction

Programming robots to execute real-world tasks is very challenging and time consuming. Approaches that rely on search-based planners work for tasks involving manipulation of rigid objects without significant dynamics. However, these approaches do not work well on manipulation tasks involving deformable materials and/or fluids due to the important role of the dynamics in task success. However, many routine tasks in the industry, and in our daily lives, involve such complications. In addition, compliant joints−typically found in recent robots like Baxter−make it difficult to specify the task based on purely geometric descriptions.

Imitation learning (IL) [1], an alternative to the traditional approach, enables humans to train robots in new tasks without being familiar with the details of robot operation. The IL approach is particularly suited to compliant manipulation tasks as it allows the humans to provide demonstration data by directly enacting examples of how to do the task. This is in contrast to teleoperation based or kinesthetic demonstrations, typical of the Learning from Demonstration approach, which make it cumbersome for the human to convey their skills to the robot.

Humans often need to perform challenging tasks multiple times in order to be able to perform them at the acceptable level of performance. Typically, under motor challenge, human performance is highly contaminated by errors during early learning stages. In-turn, throughout multiple trials humans use this motor error to adapt their neural command in order to learn the proper motor coordination. Previous approaches to imitation learning in robotics area have mainly relied on successful demonstrations ([2, 3]) with few research attempts that relied purely on failed demonstrations ([4, 5]). When transferring manipulation skills, subtle differences between the robot and the human (generally referred to as the *correspondence problem* [6]) result in noisy demonstrations from the robot perspective. Therefore, we need a robust approach to imitation learning that anticipates failures in the transfer of skills from the human to the robot and has built-in features to recover from it. Accordingly, we take a different approach to imitation learning: In addition to learning from successful demonstrations, we are also interested in learning from errors made by humans and how they recovered from these errors in subsequent trials. In the approach described in this report, we learn simple rules from human demonstration that capture how human demonstrators change parameters to transition from failed demonstrations to successful demonstrations. If the robot fails to do the task using the prescribed parameters from the transition boundary, it changes parameters using the learned rules and tries again. This capability enables it to keep trying until it succeeds.

In this report, we present our imitation learning approach for a fluid pouring task. A traditional planner would need to integrate with a computation fluid dynamics simulator to evaluate feasible path candidates [7], which would be computationally expensive to use in real-time. The experimental setup is shown in Fig. 1. The intention behind using this task is to simulate an automated production environment where the robot would be required to perform a similar task repeatedly and as fast as possible. A human first demonstrates how to successfully perform the pouring task. For a successful demonstration, the human must correctly determine how much, and how fast, to tilt the bottle in order to begin the pour. Additionally, the human must constantly track the moving container while pouring, and determine when to stop before the container exits the task workspace. These decisions will depend on (1) the table rotation speed, (2) the amount of fluid that must be poured, and (3) the initial amount of fluid in the pouring container. These variations of the task can make it challenging even for a human to perform.

2 Related Work

Imitation learning (IL) is a vast research area, especially popular in the domain of robot manipulators [8, 9, 1]. In most IL approaches, human demonstrations are used to initialize a robot's policy, which is then refined based on the robot's performance. Specific refinement approaches ranged from reinforcement learning methods [10, 11, 12] to repeated practice [13, 14] and interactive learning frameworks [15, 16].

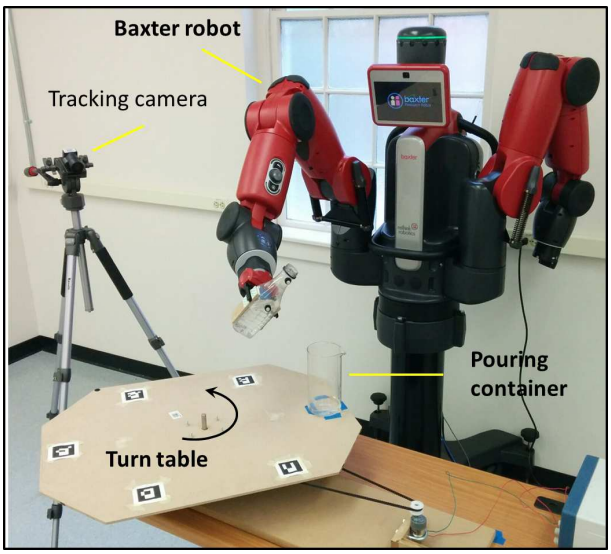

Figure 1: Physical setup used to carry out the pouring task experiments.

One of the crucial questions addressed in this context is how to compensate for suboptimal demonstration quality, typically provided by demonstrators in real-world environments. The spectrum of such suboptimal demonstrations may vary from *successful* (near perfect) to *failed* and *flawed* demonstrations. Approaches differ in terms of which of these demonstrations were used to learn the task at hand; for example, incorporating only successful demonstrations into the learning algorithm, while discarding the failed demonstrations versus learning exclusively from failures [4]. Usually, failures represent failed attempts by the human during the demonstration phase. However, this definition may change based on context [5]. Approaches also differ based on how the success/failure information is used to guide learning.

Initial work focused on learning from successful demonstrations. Akgun et al. [2] presented keyframe-based Learning from Demonstration, in which sequences of key points in the demonstrated trajectories were extracted and used to teach robot manipulators. However, the approach assumes that the demonstrator is providing optimal trajectories. The work of Khansari, et al. [3] demonstrates a method by which a dynamic system—providing directions at every point in joint/end-effector space—can be learned from multiple demonstrations and constraints imposed such that the resulting system has global asymptotic stability toward the trajectory end point. A similar approach include the dynamic motion primitives (DMP) [17, 18] which relies on mixing both linear and nonlinear systems with the linear component responsible for ensuring stability. While the original formulation used only a single demonstration to define the DMP, recent work extended it to allow many [19]. However, these approaches still assume the human is providing optimal demonstrations and suboptimal ones can only be rejected by averaging them with many others [20].

Learning from critique style approach avoids suboptimal demonstrations and achieves generalization by using instructor feedback as seen in [15, 16]. Rather than the human providing full demonstration trajectories, he or she observes the robot planning and executing its own trajectories and halts the robot motion as necessary, while providing appropriate corrections. This enables the robot to learn an approximation of the hidden optimal cost function for the task iteratively. While effective for certain tasks, the robot is unable to learn from a full example trajectory provided by a human and instead must be restricted to local learning.

Taking inspiration from how humans learn from failed demonstrations and other people's mistakes [21, 22], there have been some research attempts that enabled robots to learn from failed demonstrations. Breazeal et al. [23] presented a perspective-taking approach for learning from "flawed" demonstrations. These demonstrations do not represent failures, and are considered to be sensible from the demonstrator's perspective. However, they are ambiguous or flawed in terms of the training set required by the algorithm to generalize well. This mainly occurs due to conflicts between the robot's and instructor's perceptual beliefs of the world state. The authors developed an architecture that allowed the robot to compare either perspectives in a common reference frame to identify such conflicts and use suitable queries to clarify the ambiguities present in the demonstrations.

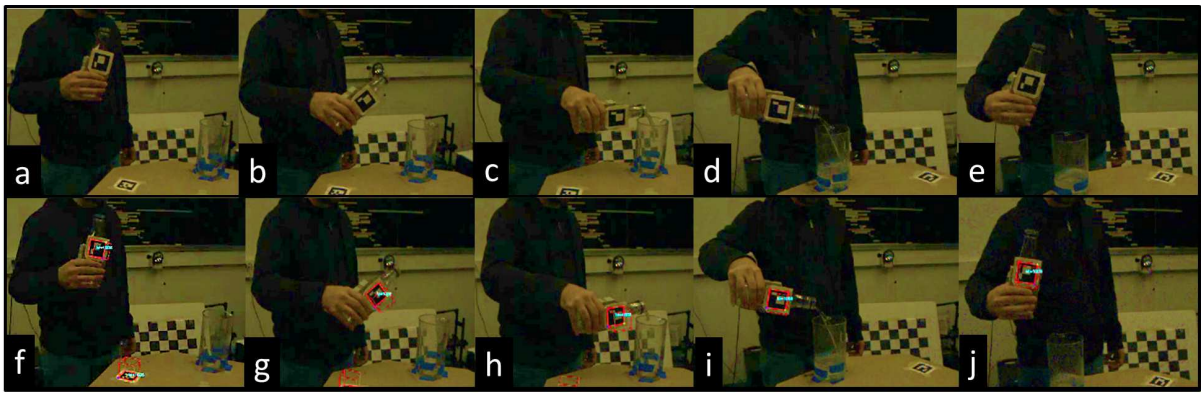

Figure 2: (a-e) Snapshots from a video footage of the human demonstration of the pouring task. (f-j) Snapshots from the corresponding video showing the visual tags detected by the tracking system.

Grollman et al. [4] developed an approach to learn only from failures. Their approach uses a Donut mixture model (DMM) framework that allows the system to explore regions of the task execution space away from where the demonstrator failed: In regions of high demonstrated variance, velocities generated are far away from the observed data, and in regions of low variance, velocities generated are closer to human's demonstrations. This strategy was successful for dynamic, single-dimensional tasks but had limited ability to scale into higher dimensions. Michieletto et al. [5] used an improved version of DMMs and applied it to failures. The meaning of failed demonstrations in this context was different in that they were not failed attempts by humans during demonstrations, but the ones that resulted in failures when the demonstrated motion was executed in the robot, caused mainly due to the differences in the morphologies of the demonstrator and the robot.

In contrast to these works on learning from failures, our approach allows the robot to learn from failures as well as successful demonstrations. The notion of using both successes and failures can be seen in few other recent works [24, 25]. Rai et al. 2013 [24] extended the DMM approach to unreliable robotic systems whose performance degraded over time (for example, due to errors introduced from motors accumulating over time). Their framework incorporated both successful and failure demonstrations to guide the search process that decreased the probability of selecting among demonstrated portions that have high variance and increased the selection probability in areas of low variance. Luo et al. [25] presented a method that used both success (preferred) and failure (non-preferred) demonstrations in their imitation learning approach to improve task performance. This is achieved by assigning success (preferred) states with higher reward values and failure (non-preferred) states with lower reward values in an inverse reinforcement learning framework. The authors tested their approach in a simple simulated car-driving robot control with reward features such as car speed, lane, and distance to other cars. Whereas, the failure information is implicitly encoded in these algorithms, we differ in how the success/failure information is used to guide the search process by identifying the parameters that caused different failures and figuring out what changes the human made to transition to a successful state.

Finally, a few imitation learning methods considered learning of the pouring task. Nemec et al. [26] applied their IL approach to learn pouring and matchbox flip-up tasks. Both successful and failure demonstrations were used by directly encoding the degree of success into the reward function. For example, when the matchbox flipped backward during the failed demonstration, the reward was assigned based on how close the box came to the upright position before flipping. The pouring task consisted of pouring equal amounts of liquid into a container from bottles of different volumes. Although this task is similar to the one chosen in our report, we consider a more complex scenario where the container is moving at varying speeds rather than being stationary. Kroemer et al. [27] developed a direct action perception framework—allowing the robot to predict afforded actions of observed objects— to learn a pouring task from single human demonstration. However, this work was focused on object affordances and how to generalize the pouring actions to different geometries of the pouring container.

3 Overview of Approach and Contribution

In addition to learning from successful demonstrations, we are also interested in learning from errors made by humans and how they recovered from these errors in subsequent trials. Every human trial is classified as either a successful or unsuccessful demonstration. Every unsuccessful demonstration is scored using a penalty score. We define a finite-dimensional parameter space to capture the essential features of the demonstrations. We can compute the transition boundary between successful and unsuccessful demonstrations using a support vector machine (SVM) classifier. This boundary represents non-dominated successful demonstrations. Theoretically, a point on this boundary prescribes parameters to be used by the robot to successfully carry out the task.

However, in practice using a point on the transition boundary does not always mean success for the robot because of the following two reasons. First, the transition boundary is constructed using a limited number of human demonstrations, and the parameters defining the space may not fully characterize the demonstrations' performances. So the constructed boundary is an approximation of the actual boundary. Second, differences in robot and human morphologies result in subtle differences between their behaviors as they try to execute a task with the same set of parameters. So when a robot tries to execute a task based on the prescribed parameters of the transition boundary, it may not completely succeed.

In our approach, we learn rules from the demonstrations that capture how the humans change parameters to transition from failed to successful trials. If the robot fails to do the task using the prescribed parameters from the transition boundary, it changes parameters using the learned rules and tries again. This capability enables it to keep trying until it succeeds. In summary, our approach gives the robot an informed set of initial parameters to try and carry out the task. It also gives the robot rules that describe how to change the parameters if the initial set of suggested parameters does not work. Therefore, the main contribution of the work is an initial investigation into how learning can be done to simultaneously take advantage of failures and successes. Two algorithms are proposed to leverage this knowledge.

4 Pouring Task

Task configuration is defined by three parameters: (1) Target pour amount p (We assume tolerance of $\pm\ \varepsilon$ around this nominal value), (2) moving container speed v, and (3) amount of fluid in the pouring container f.

The goal is to complete this task in the small time window when the container is reachable without spilling the liquid. The task is successfully completed if (1) the amount poured in the container is between $p + \varepsilon$ and $p - \varepsilon$ and (2) no fluid is spilled. If the task cannot be successfully completed, then we assign a penalty score. The penalty score is the amount of fluid that is outside of the tolerance range. Based on our initial exploration, the following four parameters need to be selected to carry out the task:

1. Container Tilt Angle α. This represents the amount the pouring container is initially tilted to start the pouring.
2. Container Tilt Angle Speed ω. This represents the average speed used in tilting the container from the upright position to the final position.
3. Post Tilting Time t_p. This represents the time from the tilting completion to task completion.
4. Final Tilt Angle α_f. This represents the final tilt angle of the pouring container at task completion.

4.1 Generating Initial Task Parameters

Let D be the set of demonstrations performed by the human. Each demonstration $d \in D$ is represented as a triple (s, g, λ), where s is the state, g is the outcome (e.g., success, or failure), and λ is the score (e.g., merit score for success and penalty score for failure). Let D^s be the set of success demonstrations and D^f be the set of failure demonstrations. State is represented as $(p, v, f, \alpha, \omega, t_p, \alpha_f)$.

4.1.1 Human Demonstrations

A set of 190 human demonstrations of the pouring task was generated. Out of these, 4 outliers and 16 invalid trials were removed. Accordingly, the demonstration set D had 170 demonstrations. This data was generated by observing four different demonstrators. Snapshots from a video recording of a sample human demonstration are shown in Figs.

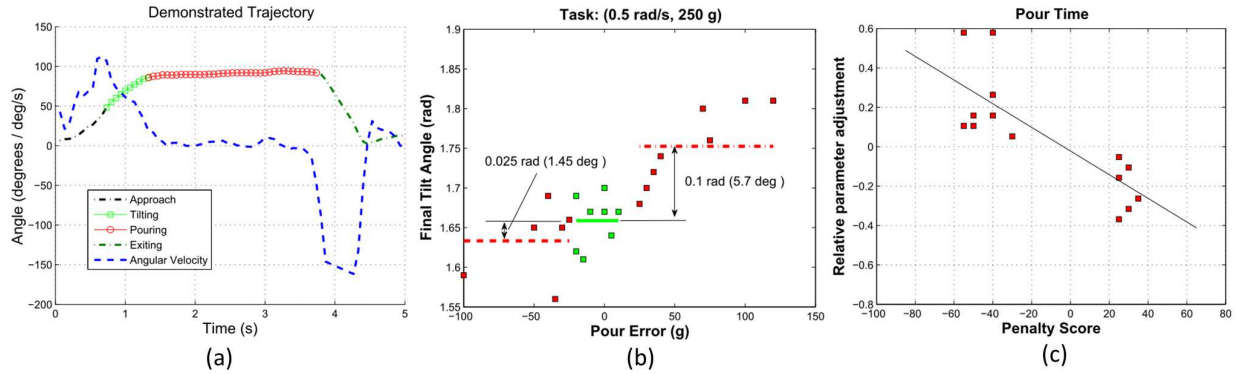

Figure 3: (a) The demonstration trajectory extracted from the video shown in terms of the tilt angle of the bottle as a function of time. (b) Plot of final tilt angle α_f as a function of pour error. A loose correlation can be seen between the parameter value and the trial performance. (c) Interpolation function giving the relative necessary change of the parameter t_p as a function of the penalty score.

2(a)-2(e). Each demonstration was classified as either a success or a failure. Out of 170 demonstrations, 93 were classified as 'success' and 77 were classified as 'failure'. Appropriate score was assigned to each demonstration.

4.1.2 Parameter Extraction

Variables p, v, and f were set for each demonstration. Variables α, ω, t_p, and α_f were extracted automatically from video recordings of human demonstrations. Multiple visual tags are attached to both the pouring bottle and the table, which are tracked by a vision system. The tag information was then used to calculate the angle that the bottle deviated from vertical. Other rotational information was ignored. This was done for each frame, generating a trajectory of the bottle's tilt over time. This trajectory was then numerically differentiated and smoothed by averaging nearby samples to provide an angular velocity trajectory as well. An example of both trajectories can be seen in Fig. 3(a).

To extract the pour task parameters, the trajectory was divided into four segments: (1) the approach to begin pouring, (2) the tilting phase, (3) the steady-state pouring phase, and (4) everything after pouring was finished. The segments were determined by thresholds on the trajectory, with tilting starting after the angle exceeds 45 degrees (ϕ_{thres}), tilting ending when the angular velocity drops below 30 deg/s, and pouring ending when the angular velocity rises above 30 deg/s to return back to vertical. These separate segments can be seen, identified by color, on the previous figure. The initial tilt angle was then specified as the average of 3 samples after tilting ended. The tilt speed was the average of velocity samples during the tilt phase. The pour time was the duration of the pouring phase, and the final angle was the average of 3 samples prior to the pour ending. In all cases, multiple samples were taken to reduce the likelihood of noise in a single sample affecting the parameter value. Finally, the table speed was also directly measured by the tracking system.

4.1.3 Training a SVM Classifier

We begin by training a SVM classifier on D. The SVM parameters and specific kernel function were selected by taking the best classification performance using 10-fold cross-validation. This resulted in use of the polynomial degree 2 kernel, with a cross-validation accuracy of 70.7%. Note that we can never expect full classification accuracy due to the reduction of the continuous demonstration to a small set of parameters.

4.1.4 Iterative Search

Given a new task configuration (p, v, f), the goal is to compute task parameters α^*, ω^* ,t_p^*, and α_f^*. We generate many states by holding (p, v, f) constant and varying α, ω, t_p , and α_f. Currently we use 10 levels for each parameter. These lead to 10000 candidate states $\mathscr{S}$. Each of the these initial states is classified as either a success or a failure using the

classifier trained using D. We delete states that are classified failure from $\mathscr{S}$. Let S^r represent the set of remaining candidate states. Now, we compute the closest distance from the states in S^r to success-states in D. We used weighted Euclidean distance between state pairs as the distance measure. Figure 3(b) shows a graph of values of parameter α_f for success (green colored) and failure (red colored) points as a function of error in pour amounts. The difference between average values of success cases and over-pour failure cases (and success and under-pour failure cases) can be used to set the weight for that parameter. Let $s \in S^r$ and let s' be a success-state in D that is closest to s. We do iterative search on the line joining s and s' to find a success-state s" that is closest to s'. Any point on the line $\overline{ss'}$ is given by:

$$q(\gamma) = \gamma \begin{bmatrix} p \\ v \\ f \\ \alpha^s \\ \omega^s \\ t_p^s \\ \alpha_f^s \end{bmatrix} + (1-\gamma) \begin{bmatrix} p \\ v \\ f \\ \alpha^{s'} \\ \omega^{s'} \\ t_p^{s'} \\ \alpha_f^{s'} \end{bmatrix} \quad (1)$$

where $\gamma \in [0,1]$. Accordingly, the search is performed by varying γ and using the classifier to check the status of states being generated during the search. From (1), note that variables p, v, and f remain constant for any value of γ. This step is performed for all states in S^r and the resulting such closest success-states are collected in the set S". Finally, we select s" $\in S$" that has the closest neighbor in D. This state is used to compute task parameters α^*, ω^*, t_p^* and α_f^*.

Algorithm 1 Algorithm to generate initial task parameters

Input: $S = \{s : s = (p, v, f, \alpha, \omega, t_p, \alpha_f)\}$,
Human demonstrations:
$D = \{(s_1, g_1, \lambda_1), (s_2, g_2, \lambda_2), \ldots, (s_n, g_n, \lambda_n)\}$, $n = |D|$
$s_i \in S$, $g_i \in \{0,1\}$ (0: failure, 1: success),
$D^s = \{(s_i, g_i, \lambda_i) : g_i = 1\}$, $D^f = D - D^s$;
$kernel \leftarrow InitializeKernel(kernel\_name, kernel\_parameters)$;
svmStruct $\leftarrow$ $svmTrain(D, kernel)$;
Initialize new task configuration $(p, v, f) \leftarrow (p_0, v_0, f_0)$;
$\mathscr{S} \leftarrow$ $GenerateCandidateStates(\alpha, \omega, t_p, \alpha_f)$;
for $i = 1 : |\mathscr{S}|$ **do**
  $g_i \leftarrow$ $svmClassify(svmStruct,\ s_i \in \mathscr{S})$
  if $(g_i == 1)$ **then**
    $s_i' \leftarrow \arg\min_{d_j \in D^s} ||s_i - d_j||$;
    $\gamma = 0$;
    while $(\gamma <= 1)$ **do**
      s_i" $\leftarrow q(\gamma)$; % From (1)
      g_i" $\leftarrow$ $svmClassify(svmStruct, s_i$")
      if $(g_i$" $== 1)$ **then**
        $ClosestDistance(s_i$") $\leftarrow ||s_i$" $- s_i'||$;
        break;
      end if
      $\gamma \leftarrow \gamma + \sigma$; % σ $<<$ 1 is a very small positive increment.
    end while
  end if
end for
return $s^* \leftarrow \arg\min_{s_i$"$} \{ClosestDistance(s_i$")$\}$;

4.2 Refining Initial Task Parameters

The robot executes the task using the parameters α^*, ω^*, t_p^*, and α_f. Let s^* be the state associated with this task execution. If the task is successful, then we stop. If the task is not successful, then these initial parameters need adjustment. For every unsuccessful demonstration in D, we ask the demonstrators as to what parameters they will change to improve performance. This is recorded for every unsuccessful demonstration. Let x be the parameter

identified by the human with an unsuccessful demonstration d. We perform line search on this parameter using the learnt SVM classifier to identify the minimum change in the value of the parameter to transition from failure to success. Let $\delta\bar{x}$ be the normalized value of target parameter change defined as:

$$\delta\bar{x} = \frac{\delta x}{x^f} \tag{2}$$

where x^f is the value of the parameter in the failed demonstration. For every parameter, identified by demonstrators, that can be varied to improve the outcome, we develop an interpolation function that expresses the normalized value of the target parameter change as a function of the penalty score. The rationale is based on our expectation that a large change in parameter value to transition to success if the penalty score associated with the failed task is high. We find the closest failed demonstration $d \in D$ to s^*. We use the parameter identified by the demonstrator in d for performing the change. We use the penalty score λ^* associated with s^* as an input to the normalized interpolation function to compute the change in the parameter. The robot tries again using the new parameter value. Currently, we stop after one round of parameter adjustment. In future, this step will be repeated until the robot succeeds.

Algorithm 2 Algorithm to refine initial task parameters

Input: $s^* = (p^*, v^*, f^*, \alpha^*, \omega^*, t_p^*, \alpha_f^*)$,
svmStruct (Trained SVM classifier from Algorithm 1),
$D^f = \{(s_i, g_i, \lambda_i) : g_i = 0\}$,
human identified parameters for each failed state
$X = \{x_i : x_i \in \{\alpha, \omega, t_p, \alpha_f\}, i = 1, 2, \ldots, |D^f|\}$;
$D_1^f \leftarrow \{(s(x_i), 0, \lambda_i) : x_i (\in X) = \alpha\}$;
$D_2^f \leftarrow \{(s(x_i), 0, \lambda_i) : x_i (\in X) = \omega\}$;
$D_3^f \leftarrow \{(s(x_i), 0, \lambda_i) : x_i (\in X) = t_p\}$;
$D_4^f \leftarrow \{(s(x_i), 0, \lambda_i) : x_i (\in X) = \alpha_f\}$;
for $i = 1 : 4$ **do**
  $n_i \leftarrow |D_i^f|$
  for $j = 1 : n_i$ **do**
    while (1) **do**
      $x \leftarrow LinearSearch(x_{ij}^f)$;
      $g_j \leftarrow svmClassify(svmStruct, s(x))$;
      if $(g_j == 1)$ **then**
        $\delta\bar{x}_{ij} \leftarrow \frac{x - x_{ij}^f}{x_{ij}^f}$;
        break;
      end if
    end while
  end for
  $interp(i) \leftarrow polyfit\left(\{(\lambda_{ij}, \delta\bar{x}_{ij}) : j = 1, 2, \ldots, n_i\}\right)$
end for
$d \leftarrow \arg\min_{d \in D^f} ||s^* - d||$;
$i \leftarrow \arg_i (D_i^f : d \in D_i^f)$; % Index of failure subset that contains d.
return $x_i^r \leftarrow x_i^* [1 + polyval(interp(i), \lambda^*)]$;

4.3 Experimental Results

Implementation of our approach was conducted with a Baxter robot and a Labvolt Model 5150 robot. In this report, we present the results using the Labvolt robot (Experiments with the Baxter robot are currently in progress).For the pouring task, the position and speed of the table was measured using the same tracking system used for demonstrations. The arm trajectory was specified simply by solving the inverse kinematics to keep the opening of the bottle above the target container. Due to constraints in the robot's abilities, the pouring phase of the trajectory was required to be discretized into two segments of minimum 1.5 seconds each, which was still sufficient to vary the pour time and keep the bottle above the target. The overall tilt of the bottle was not affected by the limits of the robot and the tilt profile was fully defined by the task parameters.

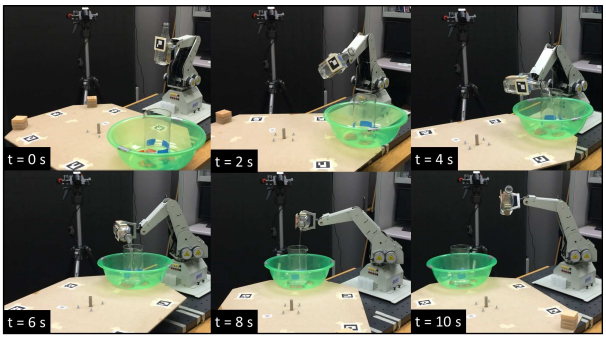

Figure 4: Snapshots from a video footage of the robot using the adjusted parameters to successfully perform the pouring task

We wanted to generate parameters for the following task configuration: $p = 300$ grams and $v = 0.34$ rad/s. We kept f (= 400 grams) fixed in all experiments). Using Algorithm 1 based on the approach described in Section 4.1, we computed the following task parameters: $\alpha^* = 1.4$ rad, $\omega^* = 1.28$ rad/s, $t_p^* = 4.1$ s, and $\alpha_f^* = 1.75$ rad.

Task execution with these parameters led to over-pour of 325 g. For the closest data point in the failed demonstration set, the human had selected pour time t_p as the parameter to improve. Algorithm 2 giving the relative change of the parameter t_p as a function of the penalty score is shown in Fig. 3(c). Note that even with the noisy data, a physically realistic fit is achieved, with an over-pour leading toward a less pronounced pouring angle. It is also passes close to the origin, providing some qualitative stability assurance that small errors do not lead to large adjustments. Then using t_p and the implied penalty score of +25, the derived interpolation function provided a relative parameter adjustment of -0.172. This provided a new pour time of 3.39 seconds. Execution of this updated set of parameters proved successful and the refinement process was halted.

5 Conclusions

We presented an imitation learning approach that allows robots to learn from how humans recover from failed attempts to perform dynamic manipulation tasks. Our algorithm allowed the robot to perform the task under variations without a complicated planning system. Experimental results showed that this approach was able to succeed at the dynamic pouring task. Future work will focus on systematic empirical evaluation with the Baxter robot and a better tracking system.

Acknowledgement

This work is supported by the Office of Naval Research under Grant N000141310597. Opinions in this paper are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[3] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, Oct 2011.

[4] D. H. Grollman and A. G. Billard, "Robot learning from failed demonstrations," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 331–342, 2012.

[5] S. Michieletto, A. Rizzi, and E. Menegatti, "Robot learning by observing humans activities and modeling failures." in *IROS workshops: Cognitive Robotics Systems, IEEE (Nov 2013)*.

[6] C. L. Nehaniv and K. Dautenhahn, "The correspondence problem," *Imitation in animals and artifacts*, p. 41, 2002.

[7] S. Saimek and P. Y. Li, "Motion planning and control of a swimming machine," *The International Journal of Robotics Research*, vol. 23, no. 1, pp. 27–53, 2004.

[8] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233 – 242, 1999.

[9] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 1371–1394.

[10] P. Abbeel, D. Dolgov, A. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 1083–1090.

[11] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[12] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machince Learning Research*, vol. 11, pp. 3137–3181, Dec. 2010.

[13] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[14] S. Calinon, P. Kormushev, and D. G. Caldwell, "Compliant skills acquisition and multi-optima policy search with em-based reinforcement learning," *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 369 – 379, 2013.

[15] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[16] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., 2013, pp. 575–583.

[17] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1398–1403.

[18] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 2009, pp. 763–768.

[19] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[20] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 409 – 413, 2006.

[21] A. N. Meltzoff, "Understanding the intentions of others: re-enactment of intended acts by 18-month-old children." *Developmental psychology*, vol. 31, no. 5, p. 838, 1995.

[22] S. C. Want and P. L. Harris, "Learning from other people's mistakes: Causal understanding in learning to use a tool," *Child development*, vol. 72, no. 2, pp. 431–443, 2001.

[23] C. Breazeal, M. Berlin, A. Brooks, J. Gray, and A. L. Thomaz, "Using perspective taking to learn from ambiguous demonstrations," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 385 – 393, 2006.

[24] A. Rai, G. de Chambrier, A. Billard *et al.*, "Learning from failed demonstrations in unreliable systems," in *Humanoids Conference*, 2013.

[25] D. Luo, Y. Wang, and X. Wu, "Discriminative apprenticeship learning with both preference and non-preference behavior," in *International Conference on Machine Learning and Applications*, 2013, pp. 315–320.

[26] B. Nemec, R. Vuga, and A. Ude, "Efficient sensorimotor learning from multiple demonstrations," *Advanced Robotics*, vol. 27, no. 13, pp. 1023–1031, 2013.

[27] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A kernel-based approach to direct action perception," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 2012, pp. 2605–2610.