Incorporating Failure-to-Success Transitions in Imitation Learning for a Dynamic Pouring Task

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

***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 Baxter robot are reported to illustrate our approach.**

I. 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 compliant manipulation tasks involving deformable materials and/or fluids due to the important role of the dynamics in task success. 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 from teleoperation based or kinesthetic demonstrations, typical to the Learning from Demonstration (LfD) approach, which make it cumbersome for the human to convey their skills to the robot.

Humans often 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 imitation learning approaches 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 compliant 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, our approach also allows learning from errors made by humans and how they recovered from these errors in subsequent trials. The notion of using both successes and failures can be seen in few other works [7], [8] (See [9] for a detailed review). Whereas the failure information is implicitly encoded in their algorithms, we differ in how the success/failure information is used to guide the search process by identifying the parameters that caused failures and figuring out what changes humans made to transition to a successful state.

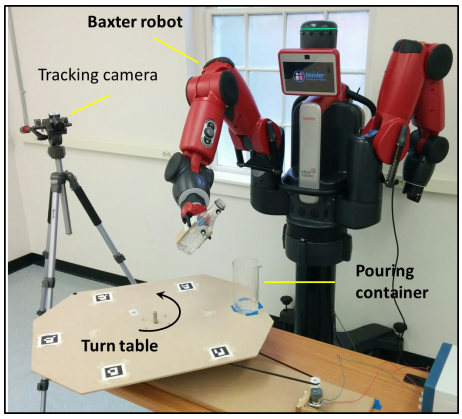

Fig. 1. Experimental setup used to carry out the pouring task.

In this paper, we present our imitation learning approach for a fluid pouring task, which involves grasping a bottle containing a fluid and pouring a specified amount of the fluid into a container placed on a rotating table. A traditional planner would need to integrate with a computation fluid dynamics simulator to evaluate feasible path candidates [10], which would be computationally expensive to use in real-time. The experimental setup is shown in Fig. 1. 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]Maryland Robotics Center, University of Maryland skgupta@umd.edu

(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. Kronander and Billard [11] addressed a pouring task consisting of grasping a full soda bottle, reaching a glass, and pouring soda into it. The authors used kinesthetic demonstrations to teach compliance (stiffness) variations to the robot: whereas, the arm must be very compliant (less stiff) while moving toward the glass (to avoid spilling), it must be stiff while pouring. Our works are closely related in that both use human demonstrations to learn compliant manipulation tasks. However, we differ primarily in two aspects: (1) learning from visual observations (typical of IL approaches) instead of kinesthetic demonstrations (typical of LfD approaches) and (2) focusing on learning how to pour the desired amount without spillage into a moving container at different speeds and using bottles of varying initial amounts of the liquid.

II. 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, 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. Learning the transitions from failure to success from the demonstration data is also where our approach differs from traditional machine learning. We do not only classify new task executions as success/failure but extract transition information from the scores of demonstrations beyond the binary positive/negative nature of an SVM classifier. 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.

III. 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.

A. 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 $(p, v, f, \alpha, \omega, t_p, \alpha_f)$.

1) Human Demonstrations: A set D of 190 human demonstrations of the pouring task was generated. Out of these, 4 outliers and 16 invalid trials were removed. This data was generated by observing four different demonstrators. Each demonstration was labeled as either a success or a failure. Out of 170 demonstrations, 93 were labeled as 'success' and 77 were labeled as 'failure'. Appropriate score was assigned to each demonstration.

2) Parameter Extraction: Variables p, v, and f were set for each demonstration. Variables α, ω, t_p, and α_f were extracted from video recordings of human demonstrations. Multiple visual tags were attached to both the pouring bottle and the table, which were tracked by a vision system. The tag information was then used to calculate the angle that the bottle deviated from vertical. This was done for each frame, generating a trajectory of the bottle's tilt over time

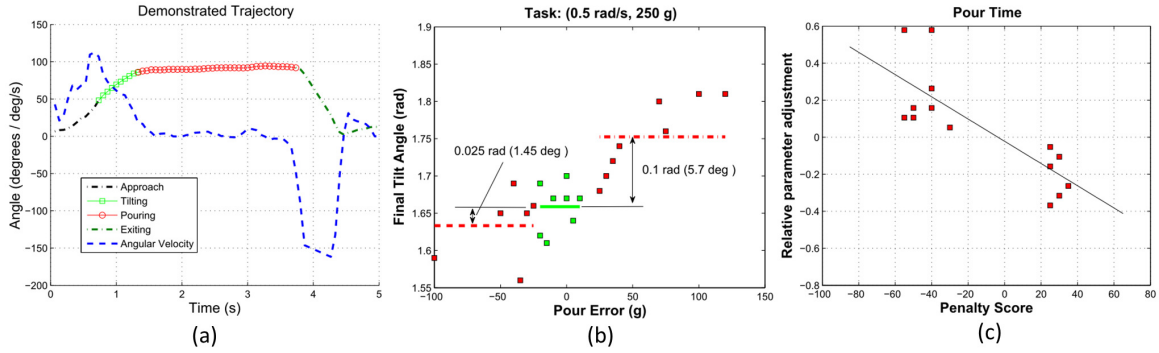

Fig. 2. (a) The demonstration trajectory extracted from the video shown in terms of the tilt angle of the bottle as a function of time. (b) The range of values for a single parameter (final tilt angle) as a function of task error shows the sensitivity to the task outcome. A loose correlation can be seen between the parameter and error values but the values of other parameters are also influential (c) Interpolation function giving the relative necessary change of the parameter t_p as a function of the penalty score.

(Example trajectories shown in Fig. 2(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. Finally, the table speed was also directly measured by the tracking system.

3) Training SVM Classifier: We first train a SVM classifier on D. The SVM parameters and kernel were selected by taking the best classification performance using 10-fold cross-validation. This resulted in use of the polynomial degree 2 kernel, with 70.7% cross-validation accuracy.

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" $\leftarrow q(\gamma)$; %
      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) 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 labeled 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 2(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'. 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^*.

B. 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. Otherwise, these initial parameters need adjustment. For every failure point in D, we query the demonstrators as to what parameters they will change to improve performance. 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. 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 $x_i^r \leftarrow x_i^* [1 + polyval(interp(i), \lambda^*)]$;

C. Experimental Results

Implementation of our approach was conducted with a Baxter robot. The pouring trajectory was discretized into two segments of 1.5 seconds each and the tilt profile of the bottle was fully defined by the task parameters. 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 III-A, 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. 2(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

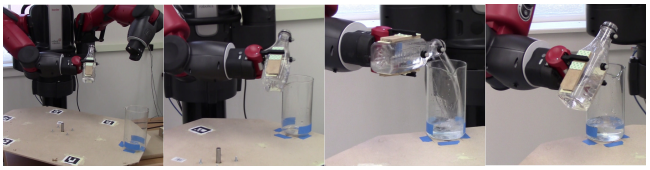

Fig. 3. Robot using adjusted parameters to successfully perform the pour

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.

IV. Conclusions

We presented an imitation learning approach that allows robots to learn from how humans recover from failed attempts to perform compliant 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 along with comparisons to related reinforcement learning approaches.

Acknowledgements

This work was supported in part by ONR grant N000141310597. The information in this paper does not necessarily reflect the position or policy of the sponsors.

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, 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 *IEEE/RSJ IROS workshop on Cognitive Robotics Systems*, 2013.

[6] C. L. Nehaniv and K. Dautenhahn, "Imitation in animals and artifacts," K. Dautenhahn and C. L. Nehaniv, Eds., 2002, ch. The Correspondence Problem, pp. 41–61.

[7] A. Rai, G. de Chambrier, and A. Billard, "Learning from failed demonstrations in unreliable systems," in *Humanoids Conf.*, 2013.

[8] 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.

[9] J. C. Langsfeld, K. N. Kaipa, R. J. Gentili, J. A. Reggia, and S. K. Gupta, "Towards imitation learning of dynamic manipulation tasks: A framework to learn from failures," *Technical Report, SBSD Lab, University of Maryland*, 2014.

[10] 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.

[11] K. Kronander and A. Billard, "Learning compliant manipulation through kinesthetic and tactile human-robot interaction," *IEEE Transactions on Haptics, DOI: 10.1109/TOH.2013.54*.