To Stir or Not to Stir: Online Estimation of Liquid Properties for Pouring Actions

Tatiana Lopez-Guevara[1,2], Rita Pucci[2], Nicholas Taylor[1], Michael Gutmann[2], Subramanian Ramamoorthy[2], Kartic Subr[2]
[1]Heriot-Watt University, [2]University of Edinburgh
{t.l.guevara}@ed.ac.uk

***Abstract*—Our brains are able to exploit coarse physical models of fluids to solve complex manipulation tasks. There has been considerable interest in developing such a capability in robots so that they can autonomously manipulate fluids adapting to different conditions. In this paper, we investigate the problem of adaptation to liquids with different characteristics. We develop a simple training task (stirring with a stick) that enables rapid inference of the parameters of the liquid with relatively inexpensive measurement equipment (standard webcams) that robots may be assumed to have access to in the wild. We perform the inference in the space of simulation parameters rather than on physically accurate parameters. This facilitates prediction and optimization tasks since the inferred parameters may be fed directly to the simulator. First, we +demonstrate that our "stirring" learner performs better than when the robot is trained with pouring actions. Further, we show that our method is able to infer properties of three very different liquids – water, glycerin and gel – and improve spillage with increased training. We present various experimental results performed by executing stirring and pouring actions on a UR10. We believe that this decoupling of the training actions from the goal task is a significant first step towards simple and autonomous learning of the behavior of different fluids in unstructured environments.**

I. Introduction

Empowering robots with a capability to autonomously manipulate liquids will lead to impact across sectors such as engineering, medicine and the service industry. Existing capabilities in industry are restricted to carefully instrumented environments where all parameters are either controlled or known precisely. For example, robots bottling wine or food products require the receptacles to be located at specific locations and are calibrated specifically using the physical properties of the products they manipulate.

A number of exciting solutions have been proposed with respect to learning to pour liquids. Recent approaches are broadly based on reasoning using simulations of the liquid [18, 10] or on optimization based on parametric assumptions (parabolic trajectory) of the liquid [13]. The physical parameters of the setup play an important role in both approaches. This includes the shapes of the pouring and receiving containers, intrinsic and extrinsic properties of the liquid, etc. Prior works have focused on inferring specific subsets of these parameters via sampling [18] or by feedback in closed-loop [19]. The latter poses a technical challenge, of having to track particles of real fluids.

A general class of methods, popularly known as *intuitive physics* [2, 1, 21, 22, 16], argues that coarse representations of physical processes are sufficient for many prediction tasks. Inspired by this approach, Lopez-Guevara et al [10] used an approximate (but real-time) fluid simulator NVIDIA Flex [11] to represent the behavior of liquids. Although this enabled fast prediction, their inference problem involves mapping real world observations to the parameter space of the approximate simulator via a cumbersome calibration step involving pouring.

In this paper, we focus on the problem of inferring the behavior of different types of liquids using simple training interactions and their observed effects. Rather than learning physical properties, we parameterize liquids according to inputs specified to NVIDIA Flex. The learning algorithm searches this parameter space online. For this, the robot stirs the liquid using a *motion pattern* and seeks simulation parameters that match the inclination of the stick in simulation against its observed values. Finally, we use the inferred parameters to predict the optimized pouring action for a given liquid and verify that it reduces spillage. The high-level contributions of this paper, in the context of pouring liquids, are that we: (1) decouple actions performed during training from the goal; (2) propose an online, autonomous calibration action; (3) achieve adaptability to different liquids.

II. Related work and contribution

There is a large body of work that address problems in using simulations for robotic control and path planning. There is also a considerable amount of work on robots learning by interacting with their environment. We restrict our review to those works that are directly relevant to our goal of learning the physical properties of liquids for manipulation.

Estimation of physical properties: A few approaches focus on estimating physical parameters such as volume [3, 12] and viscosity [4, 17], These methods exploit special measurement equipment such as RGBD cameras or tactile sensors for parameter estimation. In our context, knowledge of the physical parameters would only be useful if a high-fidelity simulation is used to optimize decision-making during manipulation of fluids. To remain practical, it is necessary to resort to approximate simulators which typically face the *model mismatch* problem since their input parameters do not coincide exactly with physical attributes such as viscosity. Different approaches have been proposed to learn simulator parameters from data [22, 8, 5, 9]. Different to [22], we do not assume a Gaussian

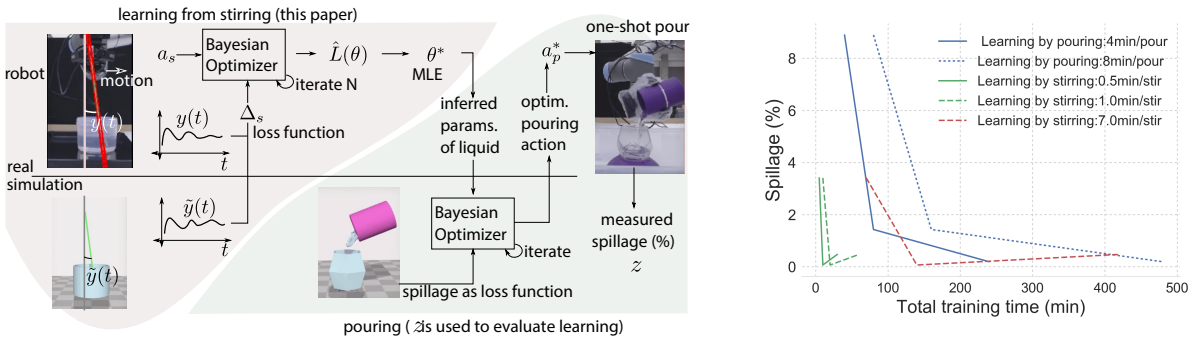

Fig. 1. *Left:* This paper focuses on learning parameters of liquids by stirring. The efficacy of learning is evaluated by executing one-shot pouring and measuring the percentage spillage z. *Right:* We compared percentage spillage achieved by our method (learning by stirring) against the baseline (learning by pouring) as a function of the time spent in traning. Stirring is both quicker (solid green curve, 0.5 min per stir) as well as more efficient compared to learning by pouring (solid blue curve, $4min$ per pour). The dashed curves are hypothetical training times if the stirring and pouring (numbers, in mins, shown in the legend).

likelihood of the real observations given the simulated ones. Rather, we learn a model of the discrepancy between real and simulated data and use it to accelerate the search using Bayesian Optimization[5][9].

Robots interacting with fluids: Existing methods that can reason about fluids, use only simulations [23, 6, 14, 15] or a combination of simulation and real observations [24, 7]. The latter categories of approaches suffer from the problem that approximate simulations deviate over time from reality complicating the envisioned effects of robotic manipulations [6]. An interesting solution, proposes to use simulations in closed-loop [19], by periodically projecting the simulated particles onto the real liquid tracked in image-space using thermal imaging. There have been a few solutions on the use of supervised learning [14, 20] and Bayesian Optimization [10] for pouring liquids.

Summary: In summary, we are inspired to combine promising directions of recent work that use supervision [20] for learning to pour from, say video of pouring actions. Simultaneously, we retain the benefits of using approximate simulation [18, 10] since it allows a generalization to a variety of manipulations. We decouple the training task from the manipulation so that it lends itself to automation, is less messy for tasks such as pouring liquids and is also more time-efficient. We avoid the need for specialized equipment such as RGBD or thermal cameras, allowing the training to be performed with two webcams which mobile robots may plausibly be equipped with. Finally, we perform parameter estimation in the space of inputs of the approximate simulator rather than physical units. This enables us to use these parameters directly for predictive tasks by supplying them to the simulator during test execution.

III. Problem Definition

Let $a_s \in \mathcal{A}_s$ denote actions performed in training, where the subscript abbreviates "stirring". Let $\theta \in \Theta$ define the parameters controlling the behaviour of the liquid in the simulation-based model. For each action a_s executed by the robot, let the observable at time t be $y(t)$, the inclination of the stick used for stirring. When the same action is executed by the simulator using the parameter θ, let the resulting inclination be $\tilde{y}_\theta(t)$. We define the observed discrepancy (for stirring) over the duration T of action a_s as

$$\Delta_\theta = \mathbb{E}\left[\int_T \left(y(t) - \tilde{y}_\theta(t)\right)^2 \, \mathrm{d}t\right] \tag{1}$$

Let $a_p \in \mathcal{A}_p$ be a pouring action and let z denote the corresponding spillage (as a percentage of the poured liquid) observed when a_p is executed. The relationship between the variables is specified in the graphical model shown in Fig. 2.

Here, we analyze the problem of inferring parameters $\theta^{*,\in}\Theta$ of the liquid, given a space of training (stirring) actions $\mathcal{A}_s$ *that are different from* the space of goal (pouring) actions $\mathcal{A}_p$. We quantify the suitability of $\mathcal{A}_s$ by measuring the percentage of liquid spilled while performing optimized one-shot pouring using $a_p^* \in \mathcal{A}_p$ obtained from $\theta^{*,\cdot}$

Assumptions: We assume that the shapes (geometry) of the containers are available, or can be estimated using sensors. Also, we rely on the robot's estimation of its end effector pose, to synchronise simulation with reality.

Inference:

Given a specific $\mathcal{A}_s$, say stirring using a motion pattern, the goal of the inference step is to estimate the best θ^* in simulation such that the discrepancy Δ_θ is minimal. At each iteration k, an action a_s is executed by the robot and in simulation using a hypothesized parameter θ. The resulting discrepancy Δ_θ^k, calculated using Eq. 1, together with the parameter θ are provided to a Bayesian Optimizer [9] that learns a regression of θ over Δ_θ using a Gaussian process, where the goal is to optimize:

$$\Delta^k(\theta) \sim \mathcal{GP}(\mu^k(\theta), \kappa^k(\theta, \theta'))$$
$$\theta^{*,k} = \underset{\theta \in \Theta}{\operatorname{argmin}}\, \mu^k(\theta)$$

An approximation of the likelihood [5] can be computed using the cdf of the standard Normal distribution Φ as (visualized in Fig. 2-*Right*):

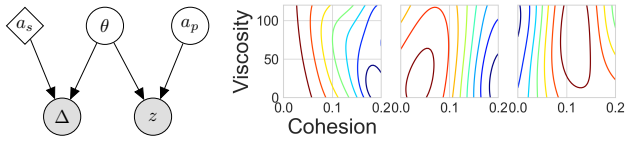

Fig. 2. *Left*: Graphical model showing relationships between variables: a_s and a_p are stirring and pouring actions respectively; Δ is the discrepancy in measured and simulated inclination; z is the measured relative spillage (percentage); and θ is the parameter that characterizes the property of the simulated liquid. By stirring, we wish to obtain a maximum likelihood estimator for $\theta^{*,\cdot}$ During pouring, we use $\theta^{*,}$ to determine an optimum a_p^* which reduces z. *Right*: Contour plots of the posterior belief on the fluid parameters after stirring water (left), glycerin (middle) and gel (right).

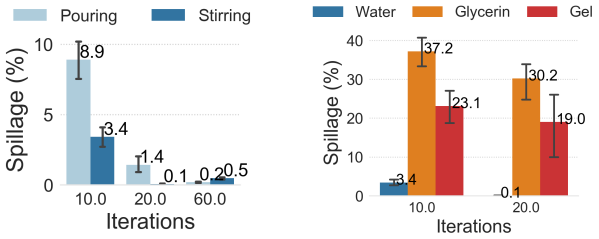

Fig. 3. Left: Effect of two calibration methods in the deployment task measured as the decrease of spillage with respect to the number of iterations. Right: Effect of the parameters inferred after performing the stirring action 10 and 20 times on three liquids.

$$\hat{L}_n(\theta) \propto \Phi\left(\frac{\epsilon - \mu(\theta)}{\sigma_n(\theta)}\right)$$

Evaluation: We quantitatively evaluate the suitability of $\mathcal{A}_s$ for the problem by measuring percentage spillage using an optimized action $a_p^* \in \mathcal{A}_p$. This is due to the lack of an existing ground truth of the parameters in the simulator given its approximate nature. Since our contribution concerns the training task, we use a pouring strategy exactly as proscribed by previous work [10]. They use a simulator to identify a_p^*, by defining the loss function to be the ratio of the spilled particles to the total number of particles simulated. The j^{th} iteration of their method therefore involves executing the simulator with action $a_p^j \in \mathcal{A}_s$ and $\theta^{*,\cdot}$ The minimization results in a_p^* after a finite number (15 in our case) of iterations. Finally, we execute a_p^* using the robot and measure the percentage of liquid spilled.

IV. Experiments and Results

A. Experimental setup

Stirring: For all our experiments, we used a UR10 robot equipped with a gripper holding a stick so that it is free to pivot at the gripping point. Before stirring begins, the stick is vertical and partly submerged in the liquid. The motion of the end effector is limited to a plane $\mathcal{P}$ parallel to the ground plane. Due to this motion, and the the resistance encountered by the stick due to the liquid, at any instant t, the stick might deviate from its vertical position to $y(t)$. The inclination is intricately dependant on the velocity of the end effector and the physical properties of the liquid and the stick. $y(t)$ is estimated using simple computer vision on the video feed from two Logitech HD Pro C920 webcams with image planes orthogonal to $\mathcal{P}$. The position of the end effector of the robot is queried, at 30Hz, and supplied to the simulator which replicates the executed action. The inclination produced in simulation at instant t is recorded as $\tilde{y}_\theta(t)$. The space of stirring actions $\mathcal{A}_s$ is discrete and determined by the stirring pattern. In this work we used a cyclic sequence, $\mathcal{A}_s = \{a_s^i\},\ i = 1, \cdots, m$ that visually follows an $m-$point star with $m = 9$.

Pouring: We replicate the one-shot pouring solution in existing work [10]. For completeness, we review their method here using our notation. The space of pouring actions $\mathcal{A}_p$ is two dimensional and continuous. The 2D space is parameterized by a constant angular velocity and the relative distance between source and target containers $a_p^i = (\omega^i, p^i)$. After 15 iterations of the optimizer over a_p given θ^*, the robot obtains an estimate for the optimal pouring action a_p^*, which it then executes. We measure the percentage of liquid spilled by the robot over 5 repetitions of the above experiment.

B. Discussion

Learning by stirring vs learning by pouring: We compared the percentage spillage z achieved by our algorithm which learns by stirring against the method proposed in [10] which calibrates by pouring using a training cup. Although it would seem intuitive that applying the same task to train must result in lower spillage under test conditions, our results indicate the contrary. Fig. 3-*Left* plots z vs N, where N is the number of iterations of the B.O. used to estimate $\theta^{*,\cdot}$ Using our stirring approach, the spillage is less than 5% even with only 10 iterations, under half the corresponding figure when the robot was trained with pouring. At $N = 20$ iterations, our approach almost achieves zeros spillage (which is lower than learning from pouring at $N = 60$ iterations).

Pouring other liquids: We observed a similar trend across three different liquids Fig. 3-*Right*: as N is increased, the spillage reduces. However, the degree of spillage is significantly higher for more glycerin and gel. On further investigation of the video and the simulator, we realized that the excessive spillage for glycerin is due to the unusually high adhesive effect that makes glycerin stick to the pouring container. Unfortunately, this adhesive behaviour cannot be modelled by the simulator on a particle-particle interaction. We conclude that the choice of the approximate simulator, combined with potentially different behavior across training and pouring actions might be a source of error during spillage. However, the capability to infer parameters within a limited gamut of expressibility is still a valuable addition to the toolkits proposed by existing methods.

Training times: Each iteration of the stirring method takes about 0.5 minutes. On the other hand, for training by pouring [10], the time taken per training iteration is 4 minutes. Even if stirring was only as efficient as pouring in terms of the number of optimization iterations, this already offers a saving

of about $8\times$ in training time. This gap is evident in the plot shown in Fig. 1-*Right*, which compares the spillage during testing resulting from the two different training approaches. The solid curves represent optimistic times taken per training task for the two approaches. The dashed curves show the maximum expected time per iteration for the two training approaches. Even if stirring took 7 mins per stirring (which is heavily exaggerated), the spillage (dashed red curve) is comparable to that achieved by "learning by pouring" (solid blue curve). The plot also shows that the total training time can be hundreds of minutes for learning by pouring.

V. Conclusion

We have presented the first supervised learning algorithm for robotic manipulation of liquids that decouples the training action (stirring) from the final task (pouring) while adapting to liquids with widely different properties. Learning by stirring is preferable to learning by pouring because it is easy to automate, it is time efficient and avoids the mess involved due to spillage. We demonstrated that stirring leads to reduced spillage for water compared to state of the art and also presented results for adapting the pouring to other liquids. We discussed the several design decisions involved, along with quantitative justification and recommendations for prospective use-cases.

References

[1] Christopher Bates, Peter Battaglia, Ilker Yildirim, and Joshua B Tenenbaum. Humans predict liquid dynamics using probabilistic simulation. In *CogSci*, 2015.

[2] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

[3] Chau Do, Tobias Schubert, and Wolfram Burgard. A Probabilistic Approach to Liquid Level Detection in Cups Using an RGB-D Camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[4] Christof Elbrechter, Jonathan Maycock, Robert Haschke, and Helge Ritter. Discriminating liquids using a robotic kitchen assistant. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 703–708. IEEE, 2015.

[5] M.U. Gutmann and J Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 17(125): 1–47, 2016.

[6] Lars Kunze and Michael Beetz. Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. *Artificial Intelligence*, jan 2015. ISSN 00043702. doi: 10.1016/j.artint.2014.12.004. URL http://www.sciencedirect.com/science/article/pii/S0004370214001544.

[7] Yoshifumi Kuriyama, Ken'ichi Yano, and Masafumi Hamaguchi. Trajectory planning for meal assist robot considering spilling avoidance. *Proceedings of the IEEE International Conference on Control Applications*, pages 1220–1225, 2008. doi: 10.1109/CCA.2008.4629665.

[8] J. Lintusaari, M.U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, 66(1):e66–e82, January 2017. ISSN 1063-5157.

[9] Jarno Lintusaari, Henri Vuollekoski, Antti Kangasrsi, Kusti Skytn, Marko Jrvenp, Michael Gutmann, Aki Vehtari, Jukka Corander, and Samuel Kaski. Elfi: Engine for likelihood free inference, 2017.

[10] Tatiana Lopez-Guevara, Nicholas K. Taylor, Michael U. Gutmann, Subramanian Ramamoorthy, and Kartic Subr. Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, pages 77–86, 2017. URL http://proceedings.mlr.press/v78/lopez-guevara17a.html.

[11] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33 (4):104, 2014.

[12] Roozbeh Mottaghi, Connor Schenck, Dieter Fox, and Ali Farhadi. See the Glass Half Full: Reasoning about Liquid Containers, their Volume and Content. *arXiv:1701.02718*, 2017. URL http://arxiv.org/abs/1701.02718.

[13] Zherong Pan and Dinesh Manocha. Motion Planning for Fluid Manipulation using Simplified Dynamics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 0, 2016. URL http://arxiv.org/abs/1603.02347.

[14] Zherong Pan and Dinesh Manocha. Feedback Motion Planning for Liquid Pouring Using Supervised Learning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[15] Zherong Pan, Chonhyon Park, and Dinesh Manocha. Robot Motion Planning for Pouring Liquids. In *The International Conference on Automated Planning and Scheduling (ICAPS)*, 2016.

[16] Vivian C Paulun, Takahiro Kawabe, Shinya Nishida, and Roland W Fleming. Seeing liquids from static snapshots. *Vision research*, 115:163–174, 2015.

[17] Hannes Saal, Jo-Anne Ting, and Sethu Vijayakumar. Active sequential learning with tactile feedback. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 677–684, 2010.

[18] Connor Schenck and Dieter Fox. Reasoning About Liquids via Closed-Loop Simulation. In *Robotics: Science and Systems (RSS)*, 2017. URL http://arxiv.org/abs/1703.01656.

[19] Connor Schenck and Dieter Fox. Visual Closed-Loop Control for Pouring Liquids. In *International Conference on Experimental Robotics (ICRA)*, 2017. URL http://arxiv.org/abs/1610.02610.

[20] Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey

Levine. Time-Contrastive Networks: Self-Supervised Learning from Multi-view Observation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July:486–487, 2017. ISSN 21607516. doi: 10.1109/CVPRW.2017.69.

[21] Tomer D Ullman, Elizabeth Spelke, Peter Battaglia, and Joshua B Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in cognitive sciences*, 21(9):649–665, 2017.

[22] Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, page 7, 2016.

[23] Akihiko Yamaguchi and Christopher G Atkeson. Differential Dynamic Programming for Graph-Structured Dynamical Systems : Generalization of Pouring Behavior with Different Skills. In *IEEE-RAS International Conference on Humanoid Robots*, number 2, 2016.

[24] Akihiko Yamaguchi and Christopher G Atkeson. Stereo Vision of Liquid and Particle Flow for Robot Pouring. In *IEEE-RAS International Conference on Humanoid Robots*, number c, 2016.