Stir to Pour: Efficient Calibration of Liquid Properties for Pouring Actions

Tatiana Lopez-Guevara\(^1,2\), Rita Pucci\(^3\), Nicholas Taylor\(^2\), Michael U. Gutmann\(^1\), Subramanian Ramamoorthy\(^1\), Kartic Subr\(^1\)

Abstract—Humans use simple probing actions to develop intuition about the physical behavior of common objects. Such intuition is particularly useful for adaptive estimation of favorable manipulation strategies of those objects in novel contexts. For example, observing the effect of tilt on a transparent bottle containing an unknown liquid provides clues on how the liquid might be poured. It is desirable to equip general-purpose robotic systems with this capability because it is inevitable that they will encounter novel objects and scenarios. In this paper, we teach a robot to use a simple, specified probing strategy – stirring with a stick – to reduce spillage while pouring unknown liquids. In the probing step, we continuously observe the effects of a real robot stirring a liquid, while simultaneously tuning the parameters to a model (simulator) until the two outputs are in agreement. We obtain optimal simulation parameters, characterizing the unknown liquid, via a Bayesian Optimizer that minimizes the discrepancy between real and simulated outcomes. Then, we optimize the pouring policy conditioning on the optimal simulation parameters determined via stirring. We show that using stirring as a probing strategy result in reduced spillage for three qualitatively different liquids when executed on a UR10 Robot, compared to probing via pouring. Finally, we provide quantitative insights into the reason for stirring being a suitable calibration task for pouring – a step towards automatic discovery of probing strategies.

I. INTRODUCTION

The development of general-purpose robots that can learn to manipulate liquids has the potential to impact multiple sectors including engineering, medicine and the service industries. Applying machine learning techniques to learn about unknown fluids is challenging due to several difficulties including sensing methods for generating data, complicated models underpinning flow and the lack of a shape or appearance prior. These hurdles are typically overcome in robotics applications either by using specific parametric approximations \([1]\), such as assuming parabolic trajectories for pouring \([2]\), or by using a fluid simulator as a model \([3]\), \([4]\). In this paper, we adopt the latter approach since it is more general.

When using simulation, it is necessary to strike a compromise between accurate and efficient (fast) models. For robotics applications that reason about liquids in closed-loop, the latter is often more important \([3]\), \([5]\). However, fast models are usually approximate and therefore introduce an additional challenge. In addition to simulation input parameters such as the shape of the containers it is also necessary to learn approximate fluid parameters with potentially no mapping to real physical parameters \([4]\). Work in the field of intuitive physics argue that humans use similar approximations to perform complex tasks \([6]\), \([7]\), \([8]\), \([9]\).

The model mismatch or “reality gap” requires the estimation of input simulation parameters as a precursor to any task-specific optimisation, i.e. if the target task is to pour a liquid optimally, the simulation parameters corresponding to the liquid in question need to be estimated first. Then, these parameters are used to determine the optimal actions to be executed by the robot. Previous work in fluid manipulation has either assumed that these input parameters are known \textit{a priori} \([10]\), \([11]\), \([12]\), \([13]\), \([14]\) or are estimated by executing the target task in a calibration stage \([5]\), \([4]\), \([15]\). However, it is often an advantage to be able to assess these parameters by performing a simpler task that does not require manual intervention (cleaning and replenishing) and that minimize the risk of damaging the robot.

We propose a method to estimate properties of unknown liquids using stirring as a simple probing strategy. We evaluate the efficacy of our probing strategy by pouring unknown
Fig. 2. This paper focuses on learning parameters of liquids by stirring. Learning parameters of liquids $\theta$ by stirring with the motion pattern $a_s$. The discrepancy $\Delta \theta$ between the real $y(t)$ and simulated $\hat{y}(t)$ time signals is obtained in real time. The efficacy of learning is evaluated by executing one-shot pouring and measuring the percentage spillage $z$.

Estimation of physical fluid properties: There are also methods that don’t assume any underlying model at all, and directly estimate real-physical parameters of liquids from data using robots. For example, to obtain the volume [22], height of the body of liquid [23], 3D shape of the container [24] and dynamic viscosity [25], [26], [27]. These methods exploit special measurement equipment such as RGBD cameras, microphones or tactile sensors for parameter estimation. In our context, knowledge of exact physical parameters is not useful since they do not correspond to their counterparts in simulation, except when using a high-fidelity simulation that involves higher computation times [12].

Contributions: The high-level contributions of this paper are that we: (1) propose a simple, online calibration action (stirring) decoupled from the target manipulation task (pouring), (2) show adaptability of the obtained estimates for pouring. We analyze the behaviour of the system in 3 liquids with a wide range of viscosities (water, glycerin and gel).

III. PROBLEM DEFINITION

Let $a_s \in A_s$ denote actions performed during calibration ("stirring"). Let $\theta \in \Theta$ define the parameters controlling the behaviour of the liquid in the simulation-based model. Let $y(t)$ describe the real observed inclination of the stick while stirring with the robot. Let $\hat{y}_\theta(t)$ be the simulated observed inclination while stirring in the simulator with parameters $\theta$. We define the observed discrepancy (for stirring) over the duration $T$ of action $a_s$ as

$$\Delta \theta = \int_0^T (y(t) - \hat{y}_\theta(t))^2 \, dt. \quad (1)$$

Let $a_p \in A_p$ be a pouring action and let $z$ denote the corresponding spillage (as a percentage of the poured liquid) observed when $a_p$ is executed by the robot.
At each iteration (real) and in simulation (sim) using a hypothesized parameter $\theta$, we evaluate their likelihood according to Eq. 3. We can also use an approximation of the likelihood [29], [30] to determine an optimum $a_p^*$, 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 $a_p^* \in A_p$ and $\theta^*$. The minimization results in $a_p^*$ after a finite number of iterations (15 in our case). Finally, we execute $a_p^*$ using the robot and measure the percentage of liquid spilled.

**IV. EXPERIMENTAL SETUP**

For all our experiments, we used a UR10 robot equipped with a gripper in combination with MoveIt [32]. Simulations were executed using NVIDIA Flex [33], running on Dell Alienware Laptop with a NVIDIA GeForce GTX 1070 Graphics card and 8GB of RAM. We used a Kern 2.5k weighting scale to measure the spillage when pouring. We used two Logitech HD Pro C920 webcams to capture orthogonal views from the stick when the robot stirs the liquid. Each stirring iteration takes around 30 seconds and it does not involve human intervention. Each pouring iteration takes between 2 and 4 minutes depending on the liquid, out of which the majority involves manual operation and cleaning. The experiments performed in this paper (including evaluation and comparison with prior work) consumed a total of about 40 robot hours and 30 person hours for supervision. We used the Engine for Likelihood-Free Inference (ELFI) [34] for the calibration process with Algorithm 1 as the model.

**A. Robot Setup**

**Stirring**: The robot’s gripper is used to hold a stick so that it is free to pivot at the gripping point (see Fig. 1). Before stirring begins, the stick is vertical and partly submerged in the liquid. The motion of the end effector is limited to a plane parallel to the ground plane. Due to this motion, and the resistance encountered by the stick due to the liquid, at any instant $t$, the stick might deviate from its vertical position by $y(t)$. The inclination is intricately dependent on the velocity of the end effector and the physical properties of the liquid and the stick, $y(t)$ is estimated using a Hough Line detector with OpenCV [35] on the video feed from 2 webcams with image planes orthogonal to $P$. We average the discrepancy across views to obtain $\Delta_\theta$. The stick is wrapped

**Algorithm 1: Inference Model ($k^{th}$ iter).**

*Input*: Stirring action $a_s$  
*Output*: Discrepancy $\Delta_\theta$

1. Sample from prior $\theta^k \sim \mathcal{U}([\theta_{min}, \theta_{max}])$
2. Get sim obs. $\tilde{y}_k(t) = \text{NvFlex}(a_s, \theta^k)$
3. Get real obs. $y(t) = \text{Robot}(a_s)$
4. Compute discrepancy: $\Delta_\theta$ using (Eq. 1)
5. Return $\Delta_\theta$

**Fig. 3.** Graphical model showing relationships between variables: $a_s$ and $a_p$ are stirring and pouring actions; $\Delta_\theta$ is the discrepancy in real and simulated inclinations; $z$ is the measured relative spillage; and $\theta$ is the parameter that characterizes the shared property of the simulated liquid. By stirring, we wish to obtain a maximum likelihood estimator for $\theta^*$. During pouring, we use $\theta^*$ to determine an optimized $a_p^*$ which reduces $z$.

**Assumptions:** We assume that the configuration of the scene including the geometry of the containers and the initial positions are available, or can be estimated using sensors. Also, we rely on the robot’s estimation of its end effector pose, to synchronize simulation with reality.

**Inference:** Given a specific $A_s$, say stirring using a given motion pattern, the goal of the inference step is to estimate the best $\theta^*$ in simulation such that the discrepancy $\Delta_\theta$ is minimal. The optimization consists of $k = 1 : N$ iterations. At each iteration $k$, an action $a_s$ is executed by the robot (real) and in simulation (sim) using a hypothesized parameter $\theta$. The resulting discrepancy $\Delta_\theta$, calculated using Eq. 1 together with the parameter $\theta$ are provided to a Bayesian Optimizer that learns a regression of $\Delta_\theta$ over $\theta$ using a Gaussian Process [28]:

$$\Delta_\theta \sim \mathcal{GP}(\mu^k(\theta), \mathcal{K}^k(\theta, \theta'))$$, \hspace{1cm} (2)

An approximation of the likelihood [29], [30] can be computed using the cumulative density function (CDF) $\Phi$ of the standard Normal distribution and a threshold $\epsilon$ as:

$$\hat{L}^N(\theta) \propto \Phi \left( \frac{\epsilon - \mu^N(\theta)}{\sigma^N(\theta)} \right)$$, \hspace{1cm} (3)

The goal is to determine the maximum likelihood estimator MLE as:

$$\theta^* = \arg\max_{\theta \in \Theta} \hat{L}^N(\theta)$$ \hspace{1cm} (4)

In addition to the MLE, the likelihood function (Fig. 8-Left) can also be used to compute samples from the posterior via a Hamiltonian Monte Carlo technique [31], by first generating samples from the prior distribution $\theta \sim \mathcal{U}([\theta_{min}, \theta_{max}])$ and evaluating their likelihood according to Eq. 3.
in bright green paper to reduce the error of the estimates of \( y(t) \). The light levels were kept similar across experiments to avoid issues with the detection. 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 \( \tilde{y}_p(t) \). The space of stirring actions \( A_s \) is discrete and determined by the stirring pattern. In this work we used a continuous sequence, \( A_s = \{ a_j \}, \ j = 1, \cdots, n \) that visually follows one of three: 9-point star, circular, or random patterns. Each action \( a_j \) corresponds to the desired position of the robot’s end effector.

**Pouring:** We replicate the one-shot pouring solution from [4]. For completeness, we review their method here using our notation. The space of pouring actions \( A_p \) 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 \( \theta^* \), the robot obtains an estimate for the optimal pouring action \( a_p^\ast \), which it then executes. We measure the percentage of liquid spilled by the robot over 5 repetitions of the above experiment.

**B. Model Sensitivity**

We started with the span of six input variables to the simulator as the whole parameter space \( \Theta \in \mathbb{R}^6 \) of liquids that can be handled by our system. We denote each parameter as \( \theta = (\theta_{ad}, \theta_{bu}, \theta_{co}, \theta_{st}, \theta_{vi}, \theta_{vo}) \) where the six variables are called (in NVIDIA Flex) adhesion, buoyancy, cohesion, surface tension, viscosity and vorticity. Using the interquartile range (IQR) on samples from the posterior along the dimensions of each \( \theta \). We used the No-U-Turn Sampler algorithm [31] with 4 chains with 1k samples each. The samples were rescaled between 0 and 1 before computing the statistics. Fig. 5 show the IQR for the parameters (cohesion and viscosity), motion patterns (star, circular, random) and liquids (water, glycerin, gel). Of the three motion patterns generated, we found the star pattern to achieve the most confidence in the inferred parameters and therefore is the one we chose to evaluate on the pouring task. We also report the obtained MLE estimates \( \theta^\ast \) across \( r = 5 \) repetitions in Table. [4] computed using (Eq. 4). These values

![Fig. 4. Model sensitivity analysis for six NvFlex parameters affecting liquid behaviour in simulation for (Left) Stirring using different patterns (star, circular, and random) and (Right) Pouring. We are interested in the shared parameter between tasks (Middle). Parameter names are abbreviated as (ad: Adhesion, bu: Buoyancy, co: Cohesion, st: Surface Tension, vi: Viscosity, vo: Vorticity).](image-url)
correspond to the simulation parameters that were used to evaluate pouring in the next 2 sections. The high IQR reported for viscosity reflects the unreliability of such parameter for estimation. This can also be seen in the likelihood contour of Figure 8 and on the variable MLE estimates along such dimension.

**Calibration by stirring vs by pouring:** We compared the percentage spillage $z$ achieved by our algorithm which calibrates by stirring against the method proposed in [4] 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. 6 plots $z$ vs $N$, where $N$ is the number of iterations of the Bayesian Optimizer (B.O.) used to estimate $\theta^*$. 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). We believe the difference between pouring and stirring at 60 iterations is caused by randomness in the simulator.

**Pouring other liquids:** We observed a similar trend across three different liquids (Fig. 7): as $N$ is increased, the spillage reduces. However, the degree of spillage is significantly higher for glycerin and gel. We empirically noticed that this is due to adhesion effects, making the liquids stick to the pouring container in the real world (Fig. 7 Right). Unfortunately, the adhesion parameter did not have any effect in simulation, creating a strong model mismatch, both in stirring and pouring. We believe that the choice of the approximate simulator is the 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.

**Iterations vs repetitions in calibration:** We use the term iterations to refer to the number of steps performed by the optimizer. At each iteration, it samples an action, executes a simulation (yielding $\hat{y}_\theta(t)$), executes the action on the real robot (yielding $y(t)$) and uses $\Delta_\theta$ as the evaluation of the loss function for that iteration. At the end of $N$ iterations, we have a single estimate for $\theta^*$. Due to the stochasticity inherent to the process, $\theta^*$ is a single realization of the optimum inferred

---

1 Due to the current COVID-19 situation we were unable to run additional experiments on calibration by pouring to validate this.
parameters. To robustify this estimate, we repeat the whole experiment by performing $N$ iterations again to yield another estimate of $\theta^*$. We then average these estimates, across repetitions, to obtain the final $\theta^*$. So performing $r$ repetitions with $N$ iterations each requires $rN$ steps of the Bayesian Optimizer (B.O.), and therefore $rN$ executions of actions by the robot. We recommend $r = 5$ and $N = 20$. Optimizing this combination, so that more robust estimates are obtained for equal calibration effort, is an interesting avenue for future work.

Calibration times: Each of the $rN$ iterations of the optimizer in calibration requires one execution of the calibration task. The stirring method takes about 0.5 minutes. On the other hand, calibrating by pouring [4] takes around 4 minutes per data sample. Even if stirring was only as efficient as pouring in terms of the number of optimization iterations, this already offers an eightfold saving ($8 \times$) in time during estimation. In addition, since stirring is more efficient when smaller number of iterations are used, the savings in calibration time by switching from pouring to stirring is significant. This gap is evident in the plot shown in Fig. 9 which compares the spillage during testing resulting from the two different calibration approaches. The solid curves represent real times taken per calibration task for the two approaches. The dashed curves correspond different hypothesis of doubling the time per iteration for each stir/pour. 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 calibration time can be hundreds of minutes as opposed to calibrating by pouring.

Simultaneously sampling $A_s$ and $A_p$: We proposed an algorithm that executes $rN$ actions from $A_N$, infers $\theta^*$ and finally performs pouring. For one-shot pouring, the optimizer samples $M$ actions, executes them in simulation and uses the ratio of spillage in simulation as the loss function to generate an optimized action $a^*_p$ which when executed on the real robot results in a spillage of $z\%$. Again, since we use B.O., just as for calibrating, $z$ is a single realization of a random variable. We obtain a more robust estimate by performing $s$ repetitions (hence the error bars in all plots with spillage on the Y-axis). Thus, for each pouring task on a different liquid, the total number of actions sampled is $rNsM$. One possibility would be to reallocate effort by increasing $M$ while setting $r = 1$. That is, for each repetition of pouring we only use a single repetition of inference. This strategy performs better overall due to the joint sampling of $A_s$ and $A_p$. If calibration is being performed solely with the goal of pouring, we recommend that an $a^*_p$ be estimated for each repetition of inference ($\theta^*$).

VI. CONCLUSIONS AND FUTURE WORK

We have proposed and evaluated a new calibration experiment that decouples the calibration action (stirring) from the final task (pouring) while adapting to liquids with widely different properties. We demonstrated that stirring leads to reduced spillage for water compared to other methods. We presented results for calibrating and adapting the pouring to other liquids. Calibration by stirring is preferable to calibration by pouring because it is easy to automate, it is time efficient and avoids the mess involved due to spillage. To our knowledge, this is the first work studying liquids that range from low to high viscosity. We also discussed the several design decisions involved, along with quantitative justification and recommendations for prospective use-cases.

Our work is limited to studying the effect of a fixed set of stirring actions that were selected after careful analysis in simulation. We believe an interesting avenue for future work lies in studying how such actions can be generated automatically using information-based metrics [36], [37], such as maximizing the information gain after each stir. Another interesting direction is to relax the current assumption of knowledge about the shapes and containers and including uncertainty in the estimation of the stick configuration.

APPENDIX

<table>
<thead>
<tr>
<th>Parameter $\theta$</th>
<th>Abbrev.</th>
<th>$\theta_{\text{min}}$</th>
<th>$\theta_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adhesion</td>
<td>ad</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Buoyancy</td>
<td>bu</td>
<td>0.3</td>
<td>2.0</td>
</tr>
<tr>
<td>Cohesion</td>
<td>co</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Surface Tension</td>
<td>st</td>
<td>0.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Viscosity</td>
<td>vi</td>
<td>0.0</td>
<td>120.0</td>
</tr>
<tr>
<td>Vorticity Confinement</td>
<td>vo</td>
<td>0.0</td>
<td>120.0</td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENTS

This work is supported by the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh. The authors would also like to thank the reviewers for their helpful comments.


TABLE II
MLE estimators for each liquid, parameter and pattern. Ranges of parameters in Table II were re-scaled to [0–1].

<table>
<thead>
<tr>
<th>Liquid</th>
<th>( \theta_{cg} )</th>
<th>( \theta_{c1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Star</td>
<td>Circ</td>
</tr>
<tr>
<td>water</td>
<td>0.03±0.07</td>
<td>0.00±0.00</td>
</tr>
<tr>
<td>glycerin</td>
<td>0.16±0.14</td>
<td>0.02±0.04</td>
</tr>
<tr>
<td>gel</td>
<td>0.65±0.16</td>
<td>0.84±0.15</td>
</tr>
</tbody>
</table>

REFERENCES


