Feedback Motion Planning for Liquid Pouring Using Supervised Learning

Zherong Pan[1] and Dinesh Manocha[1]
http://gamma.cs.unc.edu/RLFluid

***Abstract*— We present a novel motion planning algorithm for pouring a liquid body from a source to a target container. Our approach uses a receding-horizon optimization strategy that considers liquid dynamics and various other constraints. To handle liquid dynamics without costly fluid simulations, we use a neural network to infer a set of key liquid-related parameters from the observation of the current liquid configuration. To train the neural network, we generate a dataset of successful pouring examples using stochastic optimization in a problem-specific search space. These parameters are then used in the objective function for trajectory optimization. Our feedback motion planner achieves real-time performance, and we observe a high success rate in our simulated 2D and 3D liquid pouring benchmarks.**

I. INTRODUCTION

Robotic manipulation of non-rigid objects such as fluids, elastic bodies, and strings is a challenging problem that arises in different applications. In this paper, we address the problem of pouring liquids, where the goal is to use a robot to pour a liquid body from a source to a target container. These tasks arise when industrial robots are used for painting, cleaning, or dispensing lubricants. Other applications include the use of service robots for cooking, cleaning, or feeding.

A key issue in such motion planning algorithms is to satisfy the liquid dynamics constraints. The liquid body can have a complex topology and undergo large deformations. The underlying solvers tend to use a large number of particles (tens of thousands) to model their motion. This results in a very high-dimensional configuration space of the liquid body and makes it hard to directly use sampling-based motion planning algorithms. Other techniques based on an optimization-based planner [1], [2] may not work well because the free-surface of a liquid body introduces non-smooth changes.

Prior planning algorithms for fluid manipulation are either based on demonstration and learning methods, or use dynamics constraints. The demonstration-based methods use example trajectories and ignore all physical constraints so that they may not generalize to new scenarios. On the other hand, methods using reinforcement learning [3], [4] can take physics constraints into consideration, but they require a problem specific training dataset for each manipulation task. Other techniques use trajectory optimization, which takes into account a full-featured liquid dynamics model [5], [6], but these techniques have a very high computational overhead. This problem can be alleviated using reduced or simplified dynamics models [7], [8], [9], where many liquid dynamics constraints are ignored, and these models are combined with open-loop planners. However, open-loop methods suffer from simulation bias.

[1] Department of Computer Science, the University of North Carolina at Chapel Hill {zherong,dm}@cs.unc.edu

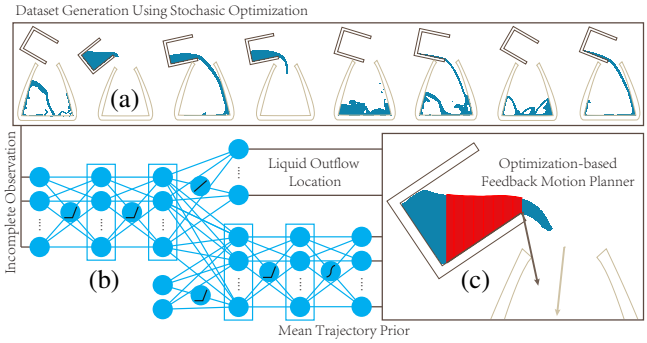

Fig. 1: An illustration of our feedback motion planning framework. From the training dataset found by stochastic optimization (a), we train a neural network that predicts liquid-related parameters: liquid outflow curve and the mean trajectory prior (b). Our online planner then computes the source container trajectory from an incomplete observation (c), e.g., the liquid heightfield in red and the moving speed of the source/target containers (the two arrows).

Main Results: We present a feedback motion planning algorithm to control the fluid flow for liquid pouring that is subject to dynamics constraints. To handle the high dimensionality of fluids, we present a learning-based approach to predict the liquid configurations based on low-dimensional features. This is combined with a receding-horizon trajectory optimization method for online planning. The main novel components of our approach include:

- A supervised learning algorithm that enables the prediction of the state of a high-dimensional liquid body using low-dimensional features. We also describe an efficient strategy to train the neural network.
- A feedback motion planning algorithm that solves a spacetime optimization problem using the receding horizon strategy. The objective function is guided by the neural network, and this significantly increases the efficiency of our planner, as we do not need to perform the costly 3D fluid simulation step at each timestep.

Our neural network is trained using a large amount of successful pouring trajectories. This dataset is automatically generated offline using a large number of random configurations of liquid pouring problems. These configurations are generated by changing the relative positions of two containers, the amount of liquid in the source container, and the speed of the target container movement. As a result, the

trained neural network is robust to environmental changes. We compute the optimal liquid pouring trajectory for each configuration using stochastic optimizations with the help of a fully-featured liquid simulator.

We have evaluated our algorithm on many new and challenging scenarios that are quite different from the training dataset. In practice, our learning-based planner achieves almost the same success rate as was obtained using the groundtruth trajectories on our training dataset. Moreover, we have also tested our planner in new and different environments, including 3D workspaces, liquids with different physical characteristics (e.g., varied the viscosities), and different container shapes. Our results demonstrate that the neural networking can be very robust when used with an optimization-based motion planner, although our training dataset uses a single low-viscous liquid material and a rectangular container shape. Furthermore, the online algorithm is very fast and takes less than 10 milliseconds on a single core of Intel CPU to plan the motion amongst dynamic obstacles.

II. ASSUMPTIONS AND PROBLEM FORMULATION

In this work, we restrict ourselves to a simulated environment. The pouring problem is formulated in Section IV. Our algorithm is composed of three components (see Figure 1).

In the preprocessing stage, we generate a large set of random liquid pouring problems. For each problem, we find a successful pouring trajectory using stochastic optimization (see Section VI). We then extract a low-dimensional feature of the liquid body and train a 4-layered neural network to predict a set of key parameters in pouring (liquid outflow curve and mean trajectory prior). We have not yet applied our framework in a real-life robotic system and currently only assume that the features can be acquired from the sensing data. However, we propose two kinds of possible liquid features to inspire future work on a real-life robotic system. Other features can also be easily used with our method.

During the online stage, the predicted parameters are used in the objective function of a receding-horizon optimization-based motion planner (see Section V). The planner determines the configuration of the source container at the next timestep by minimizing an objective function that encourages successful pouring while considering various other constraints including collision avoidance and trajectory smoothness.

III. RELATED WORKS

Our approach builds on three areas of prior work: motion planning, planning for dynamic objects, and reinforcement learning.

A. GENERAL MOTION PLANNING

A motion planning algorithm searches for a trajectory that satisfies a set of constraints (collision-free, smoothness), which may also be optimal under a given quality measure. Many early motion planners such as [10] and its descendants [11], [12], [13] consider only collision-free constraints. Unlike these methods, which tend to compute a trajectory by sampling in the space of possible trajectories, optimization-based motion planners such as [2], [1], [14] can easily take into account other constraints, such as dynamics, smoothness, etc. Many of these approaches formulate the problem as a spacetime continuous optimization. Such optimization methods have also been used for liquid transfer [6], [9] based on simplified dynamics when limited to static environments.

There is considerable work on feedback motion planning that uses refinement schemes based on feedback control laws. This can be performed using replanning [15], [16], [17] or by formulating the problem as a Markov Decision Process [18]. These ideas have been applied to high-dimensional continuous systems such as humanoid robots [19], [20]. In this work, we present such a feedback motion planning algorithm for liquid transfer.

B. PLANNING FOR DYNAMIC OBJECTS

The extension of conventional motion planning algorithms to the manipulation of non-rigid objects has been addressed in the context of virtual suturing [21], cloth folding [22], and surgical simulation [23]. It can be challenging to deal with non-rigid objects with high-dimensional configuration spaces. This is especially the case with liquid manipulation tasks, where the dimension can be as high as several million (see [6] for a detailed discussion). For certain types of fluids such as smoke and fire, optimization-based motion planning can be adapted to solve the problem by exploiting the special structure of the resulting fluid simulator [24], [25]. However, it is non-trivial to extend these methods to control liquid bodies with non-smooth, rapidly-changing free surfaces. Moreover, prior methods are designed for offline applications and computationally very costly. Previous work [9] reduced the computational cost by using a much simplified liquid model, dependent on just two variables.

C. IMITATION OR REINFORCEMENT LEARNING

Reinforcement and imitation learning have been shown to be effective in terms of controlling high-dimensional dynamic systems, e.g., a humanoid robot [19], [26]. Recently, imitation learning has been used to perform liquid manipulation using example container trajectories from a human demonstrator [27], [28], [29]. However, the learning framework in this work does not take fluid dynamics constraints into account. Moreover, trajectories of liquid body shapes from real-life experiments have to be captured and digitized to construct the dataset, which is challenging in and of itself (see [30], [31]). More recently, reinforcement learning has also been used to learn pouring of granular materials in [4], [3]. Our methods differ from these methods in that we only use supervised learning, but we combine it with trajectory optimization to enhance the robustness of our motion planner.

IV. OVERVIEW

In this section, we first introduce our formulation of the liquid pouring problem and then outline our motion planning

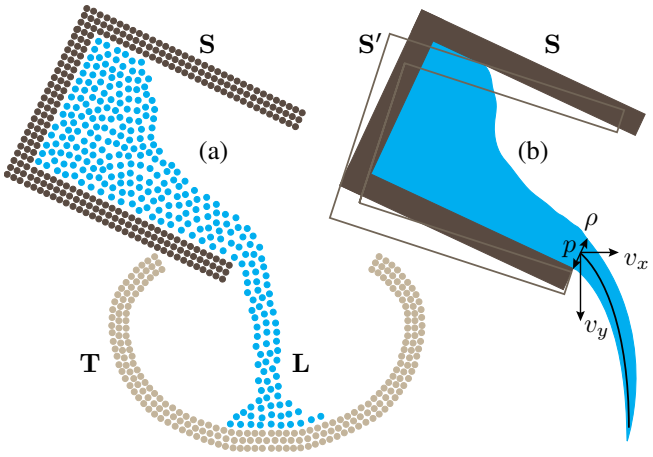

Fig. 2: (a): An illustration of our problem setting. We consider a source container $\mathbf{S}$ in dark gray, a target container $\mathbf{T}$ in light gray, and finally the liquid body $\mathbf{L}$ in blue. We use a particle-based spatial discretization for both the rigid and liquid bodies. (b): The set of learned parameters used for formulating C_o: the mean trajectory prior $\mathbf{S}'$, the parameters of liquid outflow curve p, v_x, v_y, and the outflow flux ρ.

algorithm.

A. PROBLEM FORMULATION

In each problem, we consider a source container denoted by rigid body $\mathbf{S}$, a target container denoted by $\mathbf{T}$, and the liquid body denoted by $\mathbf{L}$. Without ambiguity, we reuse these symbols to denote their spacetime trajectory $\mathbf{S}(t)$, $\mathbf{T}(t)$, and $\mathbf{L}(t)$, where t is the time index. Among these three trajectories, the liquid body trajectory $\mathbf{L}(t)$ is constrained by the Navier-Stokes equation, the governing PDE of the liquid body. Therefore, the configuration of $\mathbf{L}$ evolves as $\mathbf{L}(t + \Delta t) = f(\mathbf{S}(t), \mathbf{T}(t), \mathbf{L}(t))$, taking $\mathbf{S}(t), \mathbf{T}(t)$ as boundary conditions. Here f is a time-integration of the Navier-Stokes equation over timestep size Δt.

In practice, we define a discrete version of the problem using a particle-based spatial discretization scheme, the finite-difference temporal discretization scheme, and the same particle-based fluid simulator as [6]. As a result, all three bodies, $\mathbf{S}, \mathbf{T}$, and $\mathbf{L}$, become a set of particles, as illustrated in Figure 2 (a). Their corresponding trajectories are sampled uniformly in time with timestep size Δt. In this setting, the time-integration function f can be approximately evaluated using a discrete version of the Navier-Stokes equation.

In our problem setting, the target container trajectory $\mathbf{T}(t)$ is given. The goal of our feedback motion planner is then to determine the source container trajectory $\mathbf{S}(t)$, so that the induced liquid body trajectory $\mathbf{L}(t)$ will have the liquid body end up inside the target container $\mathbf{T}$.

V. FEEDBACK MOTION PLANNING

In order to find an entire trajectory $\mathbf{S}(t)$, our motion planner iteratively solves the following spacetime optimization problem:

$$\underset{\mathbf{S}(t+\Delta t),\cdots,\mathbf{S}(t+K\Delta t)}{\mathbf{argmin}} \sum_{1\le k\le K} C(\mathbf{S}(t+k\Delta t)) \tag{1}$$
$$C(\mathbf{S}(t)) \triangleq C_l(\mathbf{S}(t)) + C_r(\mathbf{S}(t)) + C_o(\mathbf{S}(t)),$$

over a horizon of $K\Delta t$ and then only adopts the control $\mathbf{S}(t+\Delta t)$, i.e., using the receding horizon strategy. The objective function C involves three terms: the first term C_l encourages the liquid body to fall inside $\mathbf{T}$, C_r is a regularization term that prefers smooth trajectories, and finally C_o penalizes any collisions between $\mathbf{S}$ and $\mathbf{T}$ or any other obstacles in the environment.

We can use the same C_o as [14], [6] for collision avoidance. Specifically, we define C_o as:

$$C_o(\mathbf{S}(t)) = \|D(\mathbf{S}(t))\|^2 + \|\dot{D}(\mathbf{S}(t))\|^2,$$

where D is the maximal penetration depth between $\mathbf{S}$ and any obstacle. To accelerate collision detection, each rigid object is approximated using a set of spheres. Note that we penalize both the D and $\dot{D}$ to encourage smooth movements when $\mathbf{S}$ is in the vicinity of boundaries. The regularization term $\mathbf{C}_r$ is the Laplace of the trajectory:

$$C_r(\mathbf{S}(t)) = \|\mathbf{S}(t+\Delta t) - 2\mathbf{S}(t) + \mathbf{S}(t-\Delta t)\|^2.$$

In order to define C_o such that it encourages successful liquid pouring, a naive and straightforward formulation would be to reconstruct $\mathbf{L}(t)$ from $\mathbf{S}(t), \mathbf{T}(t)$ using a liquid simulation function f, then to measure the amount of liquid that falls outside the target container and makes C_o proportional to this amount. However, this formulation has two clear drawbacks. First, reconstructing $\mathbf{L}(t)$ using liquid simulation function f is computationally costly, making it impossible for the motion planner to respond in real-time. Second, the function f involves a lot of non-smooth operators, making it difficult for numerical optimizers to find a good local minimum of Equation 1.

Our solution is to base C_o on a set of low-dimensional parameters that can be inferred from an observation of the current configuration of the liquid body $\mathbf{L}(t)$ without resorting to its predicted future configurations. This idea is like the temporal decomposition method [4]. We assume that two kinds of parameters are crucial to the task of pouring liquids. First, in most human pouring examples, the pattern is that the source container $\mathbf{S}$ moves closer to the target container $\mathbf{T}$, simultaneously increasing its turning angle so that liquid can flow out. This common pattern is encoded as mean trajectory prior $\mathbf{S}'(t)$. In addition, after liquid leaves the source container $\mathbf{S}$, the flow can be approximated as a quadratic curve, as shown in [9]. This quadratic curve could be used to guide the motion planner so that the curve is centered around the target container opening, thus avoiding spillage. In a 2D workspace, this curve is characterized by its starting point p on $\mathbf{S}$ and its leaving velocity v_x, v_y, as illustrated in Figure 2 (b). In summary, the required liquid-related parameters are $(\mathbf{S}'(t), p, v_x, v_y)$ and we then define C_l as:

$$\begin{aligned}\mathbf{C}_l(\mathbf{S}(t)) \quad = \quad & \|dist(\mathbf{T}(t), p, v_x, v_y)\|^2 \mathbf{max}(\rho, 0) + \\ & \|\mathbf{S}(t) - \mathbf{S}'(t)\|^2,\end{aligned}$$

where the first term measures the distance between liquid outflow curve and the center of the target container opening,

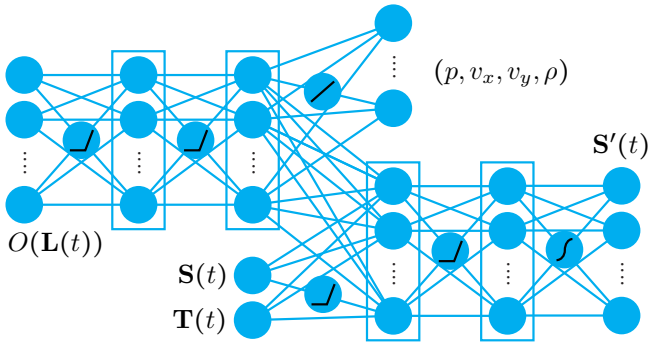

Fig. 3: Our 4-layer neural network structure for parameter estimation. The input is an observation of the current liquid configuration $O(L(t))$ and other rigid body configurations $\mathbf{S}(t), \mathbf{T}(t)$, where we assume the rigid body configurations are fully observable. We use the ReLU activation function for all internal layers and the sigmoid activation function for the output layer of mean trajectory prior $\mathbf{S}'(t)$, and linear function for the output layer of liquid outflow curve parameters. Each of the four hidden layers has 32 hidden units.

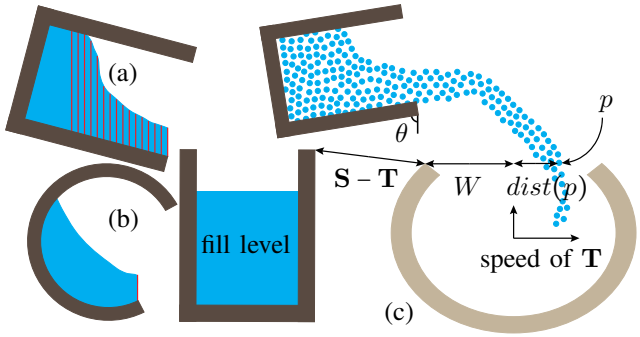

Fig. 4: The two kinds of features we use. (a): The heightfield of the liquid free-surface (red). This feature is used as groundtruth in a simulated environment. (b): The height of liquid at the lip of source container $\mathbf{S}$ (red). This feature makes it easier to apply our method on a real-life robotic system and on different container shapes. (c): The variables that define one instance of our pouring problem: relative position $\mathbf{S}$ – $\mathbf{T}$, speed of $\mathbf{T}$, and fill level. We also illustrate the variables that define our reward function R: the container opening width W and the distance from a particle to a center point $dist(p)$.

while the second term measures the discrepancy between the current source container configuration and mean trajectory prior, leading to a successful pour. Note that we added a weighting $\mathbf{max}(\rho, 0)$, where ρ is the predicted outflow flux. Therefore, we only apply the first term if liquid is flowing out. This ρ is added to the required set of parameters. These parameters $(\mathbf{S}'(t), p, v_x, v_y, \rho)$ are predicted efficiently using supervised learning introduced in Section VI.

VI. SUPERVISED LEARNING

In this section, we present our preprocessing algorithm, which uses supervised learning to predict the liquid-related parameters.

As part of the objective function, the learning algorithm needs to infer the parameters $(\mathbf{S}'(t), p, v_x, v_y, \rho)$ from the observations $O(\mathbf{L}(t))$ of a liquid body and the configurations of the two rigid bodies $\mathbf{S}(t), \mathbf{T}(t)$. The inference is accomplished with a neural network trained from a set of successful liquid pouring trajectories. See Figure 3 for a specification of our neural network.

The problem of inferring mean trajectory prior $\mathbf{S}'(t)$ from the current configuration makes our neural network a control policy representation that can be optimized using reinforcement learning as in [4], [32]. Optimized this way, the policy will predict $\mathbf{S}'(t)$ such that it can be realized directly. Instead, only supervised learning is used in our work, so the learned policy may suffer from simulation-bias issues and have a limited ability to generalize to new environments. However, we can solve this problem by plugging the predicted $\mathbf{S}'(t)$ into Equation 1 to further adjust the predicted result.

The observation function O, or the input feature for the neural network, can be defined in several ways depending on the available sensors in a certain application. In this work, we consider two kinds of observations. In a simulated environment, we use the heightfield of the liquid surface (which can be easily computed from an exact geometrical representation of $\mathbf{L}$, such as a set of particles). However, the dynamics of the liquid body are captured largely by the velocity field instead of its shape. To recover this information, we maintain a short memory of the heightfield over the past 4 frames as input to the neural network so that velocity information can be recovered by finite difference. This is illustrated in Figure 4 (a). Longer-term memory can also be recovered using, e.g., a recurrent neural network or LSTM. However, according to the definition of velocity as the derivative of positions, only the most recent memory is needed and these structures are therefore unnecessary. In a real-life robotic system, acquiring the heightfield may be very difficult. As a result, we also experimented with a simplified feature, which is the height of a liquid surface only at the lip of source container $\mathbf{S}$, as illustrated in Figure 4 (b). A common drawback of the heightfield feature is that we can only represent laminar flow without internal air bubbles in the liquid. However, we can carefully design our training dataset to contain only laminar flow (see Section VII for more details).

A. TRAINING DATA GENERATION

The dataset used to train the neural network is difficult to acquire. This dataset should contain only successful pouring trajectories. Moreover, the neural network should be expressive enough to regenerate these trajectories. Previous works [33], [34] address this problem for some applications by modifying the dataset during training. [33] assumes there is an expert who can provide additional training samples for error recovery on request. However, our approach does not assume the presence of any such expert. A human demonstrator may serve as an expert, but digitizing or capturing the liquid shape trajectory can be very challenging. On the other hand, [34] assumes that the governing equation is differentiable with respect to u. This assumption does not hold due to the non-smooth free-surface changes. Moreover, the computational complexity of [34] makes it infeasible for the high-dimensional configuration space of a liquid body.

Our solution is to use stochastic optimization to automatically search for successful pouring trajectories similar to [5] in 2D workspaces. We introduce several kinds of variations so that the learned neural network can be generalized to different problems. As illustrated in Figure 4 (a), each pouring problem can be specified by three variables:

- The relative position $\mathbf{S} - \mathbf{T}$ in range $[-3, 0] \times [-3, 3](m)$.
- The constant moving speed of $\mathbf{T}$ in range $[-5, 5]^2(m/s)$.
- The liquid fill level of $\mathbf{S}$ in range $[0.3, 0.8]$, where 1 means fully filled.

To quickly find a large number of successful pouring trajectories, we design a problem specific search space and reward function. Our liquid simulator requires very small timestep size ($\Delta t < 0.01s$) to ensure accuracy, leading to a large number of timesteps. We first limit the number of variables by using spline interpolation with 6 control points for source container trajectory $\mathbf{S}(iK\Delta t/5)$, where $0 \leq i \leq 5$. In 2D workspaces, each rigid configuration of $\mathbf{S}$ consists of 2-dimensional translation and orientation $(\mathbf{x_S}, \mathbf{y_S}, \theta_\mathbf{S})$, leading to a 15 dimensional search space (the initial control point is fixed). However, we found that this is still too large of a search space in practice because the optimizer can frequently generate zig-zag trajectories, contrary to our intuitive observation of human pouring behaviour. Therefore, we further restrict the search space by observing that source container $\mathbf{S}$ is always moving closer to $\mathbf{T}$ and its turning angle is always increasing. This gives the following relationship:

$$(\alpha_i \triangleq \frac{|\mathbf{x_S} - \mathbf{x_T}|_i}{|\mathbf{x_S} - \mathbf{x_T}|_{i-1}}, \beta_i \triangleq \frac{|\mathbf{y_S} - \mathbf{y_T}|_i}{|\mathbf{y_S} - \mathbf{y_T}|_{i-1}}, \gamma_i \triangleq \frac{|\theta_{max} - \theta|_i}{|\theta_{max} - \theta|_{i-1}}).$$

We propose to search in the transformed coordinates $(\alpha_i, \beta_i, \gamma_i) \in (0, 1]^3$ using the CMA-ES algorithm [35]. Although this is still a 15 dimensional search space, much fewer random samples are needed in each iteration with such a transformation. Finally, we use the following reward function for CMA-ES optimization, which encourages particles to pass through the center of a target container's opening and penalizes spillage:

$$R = \sum_p R_p \quad R_p = \begin{cases} \frac{W - dist(p)}{W}, & \text{if } dist(p) < W \\ -100, & \text{otherwise} \end{cases}$$

where p loops over all particles, $dist(p)$ is its distance to the center of the opening of $\mathbf{T}$, and W is half the width of the opening, as illustrated in Figure 4 (b).

After we find the optimal $\mathbf{S}(t)$ in our search space, we extract the groundtruth observation $O(\mathbf{L}(t))$ and label $(\mathbf{S}'(t), p, v_x, v_y, \rho)$ for each spline interpolated timestep. To extract the quadratic curve parameters, we use the same greedy quadratic curve fitting method as [6]. Finally, our neural-network outputs the transformed coordinates $(\alpha_i, \beta_i, \gamma_i)$, instead of $(\mathbf{x_S}, \mathbf{y_S}, \theta_\mathbf{S})$, which fit in the range of sigmoid activation function.

VII. ANALYSIS AND RESULTS

In this section, we evaluate the online and offline phases in Figure 1 from different aspects.

Quality of Dataset: All the trajectories in our dataset reside in 2D workspaces. The computational complexity of generating these trajectories is $\mathcal{O}(I \times R \times K \times P)$, where I is the number of iterations needed in each CMA-ES optimization, R is the number of random samples used in each CMA-ES iteration, K is the number of timesteps in each fluid simulation, and finally P is the number of trajectories $\mathbf{S}(t)$ we want to generate. In all the optimizations, we set $I = 200$, $R = 30$, $K = 500$, and $\Delta t = 0.01(sec)$.

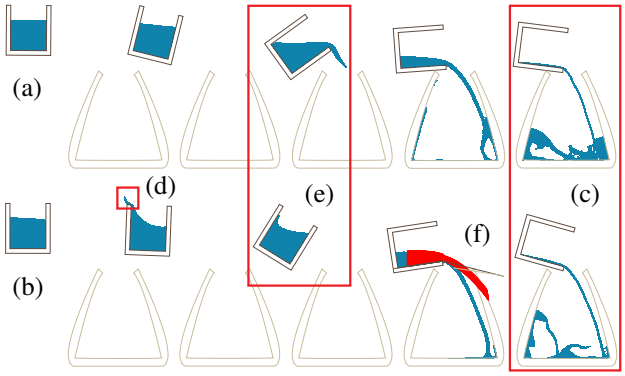

Fig. 6: We illustrate an exemplary trajectory of TRANSFER+FOLLOW (a) and TRANSFER+ZERO (b). On convergence of CMA-ES optimization, the liquid flow is well centered around the opening of $\mathbf{T}$ (c). TRANSFER+ZERO encourages spillage at an early stage of pouring (d), so that $\mathbf{S}$ must move and turn slowly (e). For each timestep in each trajectory, we extract the groundtruth water heightfield and liquid outflow curve as in training dataset (f).

In other words, each trajectory lasts for $K\Delta t = 5(sec)$. The computational complexity of dataset generation is also proportional to the overhead of performing liquid simulation, i.e., evaluating function f, which in turn is proportional to the number of particles. In 2D workspaces, we use approximately 10^5 particles and each evaluation of f takes $2.5(sec)$ while the number of particles in 3D workspaces is 10^7 and each evaluation takes $345(sec)$. Therefore, generating a 3D dataset is orders of magnitude more expensive than generating a 2D dataset.

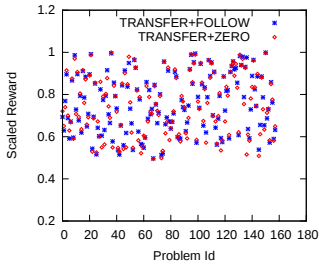

Fig. 5: The scaled reward function $R/\#Particle$ for a set of transfer problems in TRANSFER+FOLLOW (blue) and TRANSFER+ZERO (red). These values are all positive, meaning that very little spillage happens and particles are well centered around $\mathbf{O_T}$.

We generated two datasets named TRANSFER+FOLLOW and TRANSFER+ZERO, each containing $P = 1000$ successful pouring trajectories. We first sample the fill level with interval 0.12 and then select 200 relative positions $\mathbf{S} - \mathbf{T}$ and select a moving speed of $\mathbf{T}$ for each fill level using uniform random sampling in the given range. The generation of TRANSFER+FOLLOW and TRANSFER+ZERO involves a total of 6×10^9 evaluations of f. TRANSFER+FOLLOW and TRANSFER+ZERO differ in their initial liquid configuration. In TRANSFER+FOLLOW, the initial liquid velocity follows that of $\mathbf{S}$, which is typical if the liquid has moved with $\mathbf{S}$ for a certain distance and reached equilibrium. However, in TRANSFER+ZERO, the initial liquid velocity is zero, which is typical if we start the transfer from a stationary scenario. Problems in TRANSFER+ZERO are considered more difficult than those in TRANSFER+FOLLOW, as moving $\mathbf{S}$ too quickly will lead to spillage and thus negative reward. These two datasets can be downloaded at our project page and are illustrated in Figure 6.

Figure 5 shows the distribution of scaled reward function $R/\#Particle$ in TRANSFER+FOLLOW and TRANS-

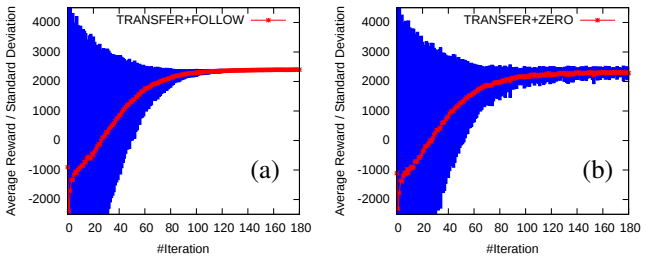

Fig. 7: The average convergence history of the CMA-ES algorithm over 1000 problems. This algorithm converges equally well for TRANSFER+FOLLOW (a) and TRANSFER+ZERO (b).

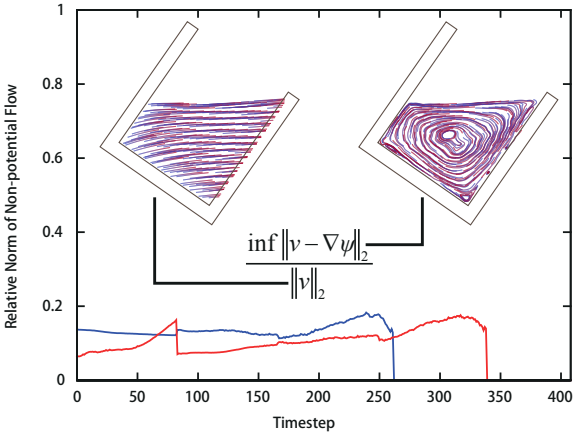

Fig. 8: We picked 100 random trajectories in TRANSFER+FOLLOW (blue) and TRANSFER+ZERO (red) datasets, and visualized the temporal change of the relative strength of the vortical velocity component (the numerator $\mathbf{inf}\|v-\nabla\psi\|_2$). Since this value is always less than 0.2, the flow is very close to a potential flow ($\mathbf{inf}\|v-\nabla\psi\|_2 = 0$). Since problems in the TRANSFER+FOLLOW dataset are less difficult, the pouring is usually completed faster, which is consistent with the early termination of the blue curve.

FER+ZERO. Figure 7 shows the CMA-ES convergence history. These figures show that our stochastic optimization algorithm can efficiently find successful transfer trajectories. Furthermore, we need to verify that the generated trajectories always transfer liquid using laminar flow, rather than turbulent flow, so that a heightfield feature can represent the shape of $\mathbf{L}$. We note that the velocity field of a laminar flow should have no internal vortex. Therefore, we compute the vortical velocity component and plot its strength relative to the original velocity field in Figure 8. This figure clearly shows that the trajectories in both TRANSFER+FOLLOW and TRANSFER+ZERO use only laminar flow.

Accuracy of Neural Network: Our neural network serves two purposes, which are verified by separate experiments.

We first verify the accuracy of the liquid outflow curve predictor by testing it on 100 randomly selected trajectories in TRANSFER+FOLLOW/TRANSFER+ZERO, i.e., we test it using the training dataset. Next, we test it on 100 new trajectories, $\mathcal{P}$, held out from training data. Figure 9 (a) plots the average accuracy against the turning angle α. This plot reveals that our predictor has low accuracy at a small turning angle (up to 0.46% at $40° \pm 10°$). The average relative accuracy throughout the pouring process is 15% using the heightfield feature, Figure 4 (a), and 24% using the height at lip feature, Figure 4 (b). At a small turning angle, using

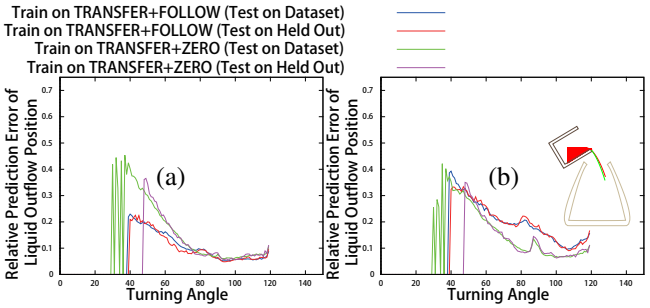

Fig. 9: Relative error of predicted liquid outflow curve parameters ($\|v_x, v_y, p\|^2$) using the heightfield feature (a) and the height at lip feature (b). The error is plotted against the turning angle of $\mathbf{S}$.

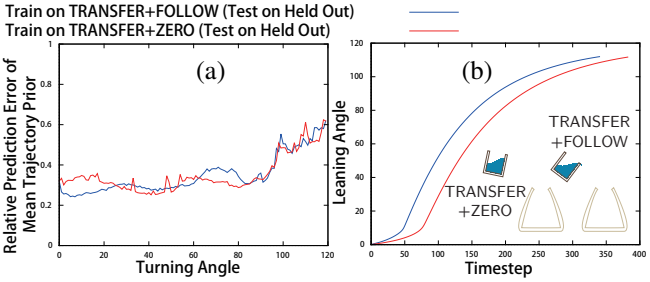

Fig. 10: (a): The relative error of predicted $\mathbf{S}(t)$ plotted against the turning angle. (b): The predicted turning angle of $\mathbf{S}$ against timestep index; the mean trajectory prior predictor trained using TRANSFER+ZERO dataset (red) learns to turn $\mathbf{S}$ slowly to avoid spillage, compared with the predictor trained using TRANSFER+FOLLOW dataset (blue). We also show such difference using one exemplary frame of a testing problem.

the height at lip feature increases the error by 36% for TRANSFER+FOLLOW dataset, while the performance is almost the same for TRANSFER+ZERO dataset. At a larger turning angle, using the height at lip feature only increases the error by 10%, at most.

We further validate the accuracy of the mean trajectory prior predictor. Figure 10 (a) illustrates the relative accuracy of $\mathbf{S}(t)$ compared with the groundtruth. The error is higher than that of the liquid outflow curve predictor. One of the main purposes of our online feedback motion planner is to compensate for this error. In addition, we conduct another experiment to see if our predictor has the ability to avoid spillage. Since we know that TRANSFER+FOLLOW does not model spillage but TRANSFER+ZERO can model spillage, we plot the predicted temporal change of α using TRANSFER+FOLLOW and TRANSFER+ZERO as the training dataset. Our predictor trained using the TRANSFER+ZERO dataset prefers a lower $\dot{\alpha}$ at small α, which implies that our network can learn the ability to avoid spillage.

Performance of Feedback Planner: In the online phase, we solve Equation 1 using a horizon length of $1.25(sec)$, with $K = 25$ and $\Delta t = 0.05(sec)$. This requires querying the neural network 25 times and performing an optimization using the LBFGS optimizer, which is very efficient due to the small size of our neural network and the optimization formulate. We tested our feedback controller on 30 new problems. For each testing problem, we experimented with three different container shapes, as illustrated in Figure 11

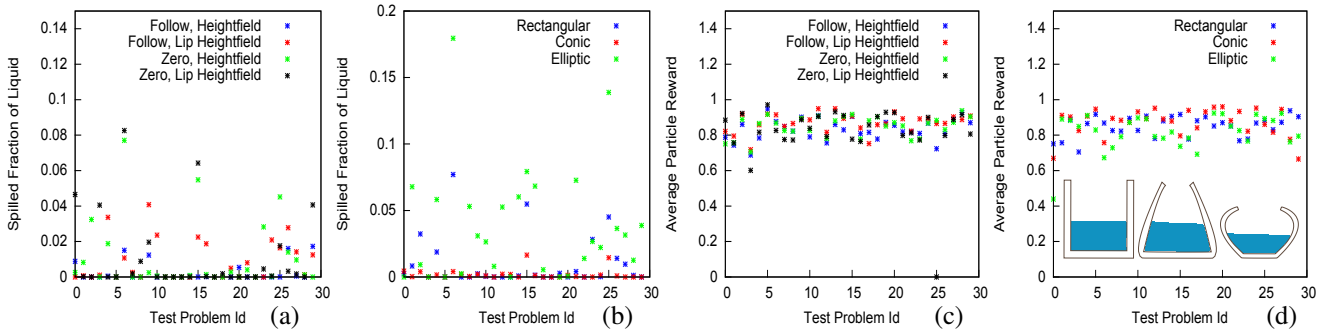

Fig. 11: Performance of the feedback planner. (a): The fraction of spilled particles using rectangular container $\mathbf{S}$, two kinds of datasets and features. (b): The fraction of spilled particles using the height at lip feature and three different container shapes. (c): Average reward of particles that fall into $\mathbf{T}$ in experiment (a). (d): Average reward of particles that fall into $\mathbf{T}$ in experiment (b).

(d). We use the same set of 30 experimental problems, which are not covered in the datasets. If the training is accomplished using TRANSFER+ZERO dataset, then we set the initial velocity to be zero and vice versa. For experiments using new container shapes, we use TRANSFER+ZERO dataset.

The performance of our planner is summarized in Figure 11, where we plot the spilled fraction of liquid and the average reward. Note that the average is taken over particles that fall into $\mathbf{T}$ (i.e., excluding the spilled particles), which is different from Figure 5, where the average includes negative values (spilled particles). We can thus analyze spillage avoidance and the accuracy of liquid flow, respectively. From Figure 11 (a), we can see that the fraction of spilled particles is less than 5% in 28 of 30 problems for both features in Figure 4. We also see that using the lip height feature increases the spillage by 4%, at most, and 0.32% on average. From Figure 11 (b), we find that, although our datasets use only rectangular containers, the spillage fraction is also very small if we test on a conic container, which results in a spillage fraction over 5% for only 2 problems. However, some containers, such as the conic container, encourage spillage and we have seen over 5% spillage in 10 problems. From Figure 11 (c), we can see that the liquid flow is generally well centered around the center opening of $\mathbf{T}$, with a mean reward of $R_p = 0.82$. If we generalize to new container shapes, the mean reward is still over $R_p = 0.8$. However, the variance is larger especially for the conic container, as illustrated in Figure 11 (d).

Generalizing to New Problems: Our motion planner is only experimented on liquid bodies of fixed material parameters, on a 2D workspace, and with two rigid bodies. Some of these limitations can be relaxed.

First, although our dataset considers only a liquid material with low viscosity, applying it to pour more viscous liquids is possible. As illustrated in Figure 12, we experimented the planner on these liquids. As we increase the viscosity, the fraction of spillage decreases and the average reward increases accordingly. This is because viscosity suppresses turbulence flow and makes our quadratic curve assumption more accurate.

Moreover, we can also relax the assumption of 2D workspaces and apply the method to 3D workspaces. As illustrated in Figure 12 (c), this is done by applying the method to only the symmetric cross section. A drawback of this treatment is that fluid may spill from directions other than the 2D plane and our method cannot avoid such spillage.

Finally, a major benefit of the optimization-based formulation, Equation 1, is that we can naturally take other dynamic obstacles into consideration without retraining or regenerating the dataset, as illustrated in Figure 12 (b). However, current obstacle avoidance term only takes rigid bodies into consideration, and collisions between liquid and rigid bodies are not considered. As a result, when the obstacle blocks the pouring, more spillage happens.

VIII. LIMITATIONS AND CONCLUSIONS

In conclusion, we present a feedback motion planner for liquid transfer problems. Our formulation uses an optimization-based receding horizon planner, which is guided by a machine learning model that provides clues for global movements (mean trajectory prior) and local adjustments (liquid outflow curve). Our experiments show that the planning framework can achieve promising online performance and the machine learning model gains important skills such as spillage avoidance and liquid position prediction. However, the current system also introduces a series of limitations, which we discuss below. We refer readers to our extended paper [36] for a thorough discussion.

First, although the spillage fraction in our dataset is almost zero, the spillage fraction can be as high as 5% in our 30 testing problems. We believe better performance can be achieved by using imitation learning [33] instead of supervised learning. Moreover, it is inherently difficult to generalize our method to discrete material types, such as a bunch of rigid bodies or granular materials as in [28], [3]. In addition, we only consider the problem of pouring the entire liquid body from $\mathbf{S}$ to $\mathbf{T}$. Other requirements might arise, e.g., if only some of liquid is needed in $\mathbf{T}$ due to the small volume of $\mathbf{T}$. Finally, two problems need to be addressed for extension to real robotic systems. We must be able to efficiently acquire liquid-related features and we must be able to extend the entire planning pipeline to 3D workspaces.

ACKNOWLEDGEMENT

This research is supported in part by ARO Contract W911NF-14-1-0437, NSF award 1305286.

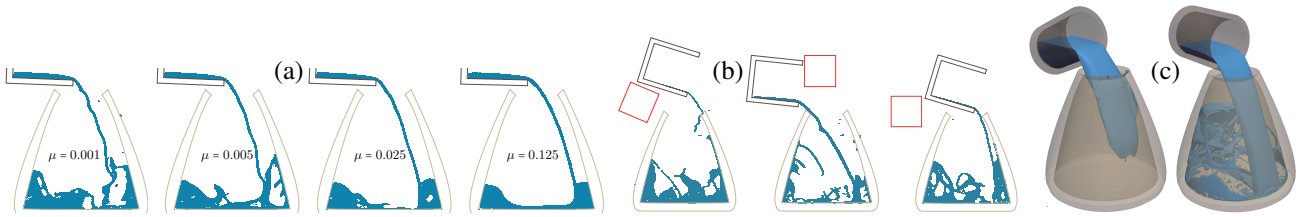

Fig. 12: (a): We tested our feedback controller on new fluid materials with higher dynamic viscosity ($kg/(ms)$). In this case, we achieved even higher online reward. (b): The controller can also account for dynamic obstacles. (c): In a 3D workspace, the controller only sees the 2D cross section.

REFERENCES

[1] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[2] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[3] A. Yamaguchi and C. G. Atkeson, "Neural networks and differential dynamic programming for reinforcement learning problems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5434–5441.

[4] ——, "Differential dynamic programming with temporally decomposed dynamics," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 696–703.

[5] Y. Kuriyama, K. Yano, and M. Hamaguchi, "Trajectory planning for meal assist robot considering spilling avoidance," in *Control Applications, 2008. CCA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1220–1225.

[6] Z. Pan, C. Park, and D. Manocha, "Robot motion planning for pouring liquids," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2016.

[7] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 107–114.

[8] M. Tzamtzi and F. Koumboulis, "Robustness of a robot control scheme for liquid transfer," in *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics*. Springer, 2008, pp. 156–161.

[9] Z. Pan and D. Manocha, "Motion planning for fluid manipulation using simplified dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2016.

[10] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] D. Wilkie, J. P. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 5573–5578.

[12] M. Stilman, "Task constrained motion planning in robot joint space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3074–3081.

[13] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, "Manipulation planning with workspace goal regions," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2009, pp. 618–624.

[14] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2012.

[15] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1243–1248.

[16] K. Hauser, "On responsiveness, safety, and completeness in real-time motion planning," *Autonomous Robots*, vol. 32, no. 1, pp. 35–48, 2012.

[17] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[18] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces," in *In Proc. Robotics: Science and Systems*, 2008.

[19] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[20] C. Park, J. Pan, and D. Manocha, "High-dof robots in dynamic environments using incremental trajectory optimization," *International Journal of Humanoid Robotics*, vol. 11, no. 02, p. 1441001, 2014.

[21] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Robotics Research*. Springer, 2016, pp. 339–354.

[22] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 6000–6006.

[23] N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O'Brien, "Interactive simulation of surgical needle insertion and steering," in *Proceedings of ACM SIGGRAPH 2009*, Aug 2009, pp. 88:1–10. [Online]. Available: http://graphics.berkeley.edu/papers/Chentanez-ISN-2009-08/

[24] A. Treuille, A. McNamara, Z. Popović, and J. Stam, "Keyframe control of smoke simulations," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 716–723.

[25] Z. Pan and D. Manocha, "Motion planning for fluid manipulation using simplified dynamics," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4224–4231.

[26] X. B. Peng, G. Berseth, and M. van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)*, vol. 35, no. 5, 2016, to appear.

[27] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1754–1760.

[28] A. Yamaguchi, C. G. Atkeson, S. Niekum, and T. Ogasawara, "Learning pouring skills from demonstration and practice," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 908–915.

[29] S. Brandi, O. Kroemer, and J. Peters, "Generalizing pouring actions between objects using warped parameters," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 616–621.

[30] H. Wang, M. Liao, Q. Zhang, R. Yang, and G. Turk, "Physically guided liquid surface modeling from videos," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 90.

[31] J. Gregson, M. Krimerman, M. B. Hullin, and W. Heidrich, "Stochastic tomography and its applications in 3d imaging of mixing fluids." *ACM Trans. Graph.*, vol. 31, no. 4, pp. 52–1, 2012.

[32] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov, "Interactive control of diverse complex characters with neural networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3132–3140.

[33] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, G. J. Gordon and D. B. Dunson, Eds., vol. 15. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011, pp. 627–635. [Online]. Available: http://www.jmlr.org/proceedings/papers/v15/ross11a/ross11a.pdf

[34] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *ICML '14: Proceedings of the 31st International Conference on Machine Learning*, 2014.

[35] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[36] Z. Pan and D. Manocha, "Feedback motion planning for liquid transfer using supervised learning," *https://arxiv.org/abs/1609.03433*.