Image Based Reconstruction of Liquids from 2D Surface Detections

Florian Richter, Ryan K. Orosco, and Michael C. Yip
University of California San Diego
{frichter, rorosco, yip}@ucsd.edu

Abstract

In this work, we present a solution to the challenging problem of reconstructing liquids from image data. The challenges in reconstructing liquids, which is not faced in previous reconstruction works on rigid and deforming surfaces, lies in the inability to use depth sensing and color features due the variable index of refraction, opacity, and environmental reflections. Therefore, we limit ourselves to only surface detections (i.e. binary mask) of liquids as observations and do not assume any prior knowledge on the liquids properties. A novel optimization problem is posed which reconstructs the liquid as particles by minimizing the error between a rendered surface from the particles and the surface detections while satisfying liquid constraints. Our solvers to this optimization problem are presented and no training data is required to apply them. We also propose a dynamic prediction to seed the reconstruction optimization from the previous time-step. We test our proposed methods in simulation and on two new liquid datasets which we open source[1] so the broader research community can continue developing in this under explored area.

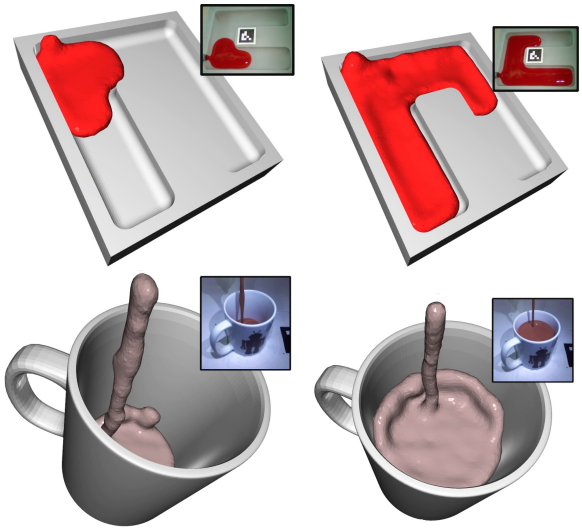

Figure 1. The top and bottom row figures shows the output of our proposed method for reconstructing liquid from an endoscopic camera and a human pouring chocolate milk into a cup respectively. Our reconstruction approach minimizes the 2D surface detection loss while simultaneously satisfying liquid constraints without the need for any prior training data. The result is an effective reconstruction technique for liquids that has been validated on simulated and real-life data as shown here.

1. Introduction

To successfully navigate in and interact with the 3D world we live in, a 3D geometric understanding is required. The importance of this requirement can be seen by the numerous advancements in reconstruction methods from cameras, which is the ideal sensor due to its information richness and cheap cost. Solutions for surface based reconstruction have been proposed for a variety of scenarios such as rigid, unknown environments [31] with dynamic objects [21]. The rigidness assumption has also been lifted to handle deformable surfaces [17, 30]. Breakthrough developments from the reconstruction community have fed into downstream applications such as robotic manipulation [43] and surgical tissue tracking [23].

Reconstruction of more complex scenes, such as fluids however remains an under explored area. Fluids, unlike rigid and deforming objects, are typically turbulent and can exhibit translating, shearing, and rotation motions [33]. The well established Navier-Stokes equations which describe fluid motion have been applied to generate effective graphic renderings of fluids [4]. The motions of fluids also differs depending on if it is a gas or liquid. Gasses are compressible and reconstruction from images has been explored [9]. Liquids, unlike gasses, are in-compressible and for everyday human interactions, rely on a container and gravity to form their shape (e.g. a mug holding coffee). By fully reconstructing liquids in 3D, automation efforts which replicate human tasks interacting with liquids can be significantly improved such as robot bar tending [44], autonomous blood suction during surgeries [16], and sewage service [42]. However, the challenge of reconstructing liquids from

[1] https://github.com/ucsdarclab/liquid_reconstruction

images remained unexplored and simplifying heuristics or end-to-end models were used to guide these automation efforts.

We propose an approach to reconstruct and track liquids from videos using minimal information. This results in the first technique to reconstruct liquids with only knowledge of the collision environment, gravity direction, and 2D surface detections. The observations are limited to 2D surface detections (i.e. binary mask) because a liquids color varies widely based on their refraction index, opacity, and environment. Furthermore, common depth sensors (e.g. Microsoft's Kinect or Intel's RealSense) will behave inconsistently due to the unknown refraction index. By limiting the observation data to only 2D surface detections, our proposed reconstruction method can be directly applied to any detected liquid and does not require any prior information on the liquid (e.g. no training data is required). To this end, our contributions are:

1. a novel optimization problem for reconstructing liquids with a particle representation which accounts for liquid constraints,

2. seeding the optimization with a dynamics prediction based on the previous time-step,

3. and a branching strategy to dynamically adjust the number of particles in the reconstructed liquid.

The complete solution only relies on sequential data and was extended with a source estimation technique to show its adaptability for future applications with liquids. To baseline our proposed method, new liquid datasets are collected and open sourced so future researchers can further develop in this under-explored area.

2. Related Works

2.1. Fluid Reconstruction

Several sensor modalities have been used historically to capture fluid flow in science and engineering. Schlieren imaging [1, 2, 6], Particle Image Velocimetry [12], laser scanners [15], and structured light [13] have all been developed for capturing fluids. These specialized sensors however are not common place and often expensive, hence making them less ideal than visible spectrum sensors. This lead to a lot of developement in the field of visible light tomography where a combination of 2D image projections of a fluid are used to reconstruct it in 3D. Recent developments in the field have effectively registered fluids with simulation based fluid dynamics [7] and require only a few camera perspectives for effective reconstruction [9, 48]. These approaches however do not consider in-compressible fluids, liquids, and only focus on gasses in free space (i.e. no collision).

2.2. Liquid Detection and Simulation Registration

While direct reconstruction of liquids has not been done before, there has been work in detecting liquids in the image frame and registering with a simulation. Pools of water have been detected for unmanned ground vehicles [34, 35], and flowing blood has been detected during surgeries for autonomous, robotic suction [37]. Liquids during a pouring task have also been detected using optical flow [45] and Deep Neural Networks [40]. The scope of this paper is on reconstructing liquids, and these detection methods could be utilized to feed into our proposed method by supplying the observations of the liquids surface. Mottaghi et al. were able to estimate a liquid's volume in a container from images directly [27]. Registration of a liquid simulation with the real world has also been conducted for robot pouring [14, 39, 41]. However, these techniques require prior information about the liquid being reconstructed, such as knowing the volume of the liquid before hand. Meanwhile in this work, we only assume prior knowledge of the gravity direction and collision environment and use a novel branching strategy to dynamically adjust the volume of the reconstructed liquid. Nevertheless, we integrated Schenck and Fox's most recent simulation registration work [41] to the best of our ability into our reconstruction approach for comparison.

3. Methods

Let $\mathbf{p}_t = \{\mathbf{p}_t^i\}_{i=1}^N$ be the set of particles in $\mathbb{R}^3$ representing the reconstructed liquid at time t. To estimate the particle locations, and hence reconstruct the liquid, we assume only knowledge of a binary masked image which identified the liquids surface, $\mathbb{I}_t$. The estimation for the particles is done by minimizing a loss between the detected surface and a reconstruction of the liquid surface from the particles, $\hat{\mathbb{I}}(\cdot)$. Written explicitly, the optimization problem is:

$$\arg\min_{\mathbf{p}_t} \mathcal{L}\left(\mathbb{I}_t, \hat{\mathbb{I}}(\mathbf{p}_t)\right) \quad \text{s.t. } \mathbf{C}(\mathbf{p}_t) = 0 \tag{1}$$

where liquid constraints, $\mathbf{C}(\cdot)$, are applied to the particles positions so they behave like a liquid. The position constraints considered here are density and collision, and a visual explanation is shown in Fig. 2. Solving position constraints and deriving velocities from them has produced stable, particle based simulations for large time-step sizes [25, 29]. Similarly, we leverage the liquid-like dynamics induced by position constraints for effective liquid reconstruction from video sequences (i.e. going from t to $t+1$).

The following methods detail our solution to the optimization problem shown in (1) and an outline is shown in Algorithm 1. First, the position constraints, $\mathbf{C}(\cdot)$, and their respective solvers are described. Second, the rendered sur-

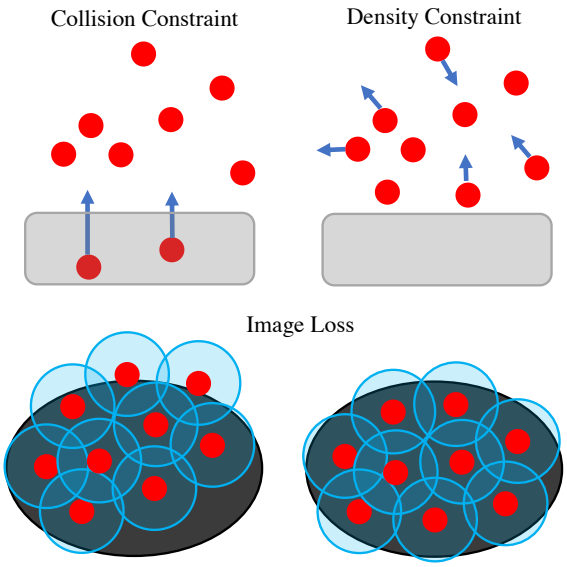

Figure 2. A visualization of solving (1) in order of top-left, top-right, and bottom where the particle locations are drawn in red. The collision constraint pushes particles out of collision (drawn in light grey), the density constraints ensures incompressibility for liquids by maintaining a constant density, and finally the image loss between the detected surface (drawn in black) and a surface rendering (drawn in semi-transparent blue) is minimized.

face, $\hat{\mathbb{I}}(\cdot)$ and its gradient with respective to the particle positions to minimize the loss is explained. The constraint solvers and surface loss gradient are applied in a projective gradient descent scheme to solve (1) as shown in lines 4 to 11 of Algorithm 1. Third, finding the number of particles, N, to reconstruct the liquid and a strategy of where to add or remove the particles is detailed. Lastly, prediction of the particles from time-step t to $t+1$ is defined to reconstruct from videos of detected liquids, $\mathbb{I}_1, \ldots, \mathbb{I}_T$.

3.1. Position Constraints for Liquid Particles

The two position constraints used to reconstruct the liquid when optimizing (1) are collision and density. The collision constraint ensures that none of the particles representing the reconstructed liquid are in collision with the scene. Let $C_c(\cdot)$ be the collision constraint for a particle, and it is expressed as:

$$C_c(\mathbf{p}^i) = \text{relu}(-SDF(\mathbf{p}^i)) \tag{2}$$

where relu$(\cdot)$ is the rectified linear unit function and $SDF(\cdot)$ is the signed distance function of the scene. The collision constraint is satisfied when it is at 0, which occurs by having all of the particles out of collision (i.e. no more negative SDF values at the particle positions).

To push the particles out of collision and satisfy the collision constraint, finite difference is used to approximate a

Algorithm 1: Reconstruct Liquid at time t

Input : Previous liquid particle positions and velocities, $\mathbf{p}_{t-1}, \mathbf{v}_{t-1}$, and image, $\mathbb{I}_t$
Output: Updated liquid particle positions and velocities $\mathbf{p}_t, \mathbf{v}_t$

// Particle Prediction
1 $\mathbf{p}_t \leftarrow \mathbf{p}_{t-1} + \mathbf{v}_{t-1}\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2$
2 **for** n_o *iterations* **do**
3 **for** n_j *iterations* **do**
  // Apply Position Constraints
4 **for** n_c *iterations* **do**
5 $\Delta\mathbf{p}_c \leftarrow solveCollision(\mathbf{p}_t)$
6 $\mathbf{p}_t \leftarrow \mathbf{p}_t + \Delta\mathbf{p}_c$
7 $\Delta\mathbf{p}_\rho \leftarrow solveDensity(\mathbf{p}_t)$
8 $\mathbf{p}_t \leftarrow \mathbf{p}_t + \Delta\mathbf{p}_\rho$
  // Minimize Image Loss
9 **for** n_i *iterations* **do**
10 $\hat{\mathbb{I}}(\mathbf{p}_t) \leftarrow renderSurface(\mathbf{p}_t)$
11 $\mathbf{p}_t \leftarrow \mathbf{p}_t + \alpha_{\mathbb{I}} \left(\partial\mathcal{L}\left(\mathbb{I}_t, \hat{\mathbb{I}}(\mathbf{p}_t)\right)/\partial\mathbf{p}_t\right)$
  // Adjust Particle Count
12 **if** *local_minima_conditions* **then**
13 $\mathbf{p}_t \leftarrow duplicateOrRemoveParticle(\mathbf{p}_t)$
// Update Particle Velocities
14 $\mathbf{v}_t \leftarrow (\mathbf{p}_t - \mathbf{p}_{t-1})/\Delta t$
15 $\mathbf{v}_t \leftarrow dampVelocityAndApplyViscocity(\mathbf{p}_t, \mathbf{v}_t)$
16 **return** $\mathbf{p}_t, \mathbf{v}_t$

gradient of (2) and the particles are moved along the gradient step. This is computed for particle $\mathbf{p}^i$ as follows:

$$\Delta\mathbf{p}_c^i = C_c(\mathbf{p}^i) \sum_{\mathbf{k}\in K} w_k SDF(\mathbf{p}^i + d\mathbf{k}) \tag{3}$$

where K is the set of finite sample directions (e.g. [$\pm$1, 0, 0], [0, $\pm$1, 0], [0, 0, $\pm$1]), w_k is the finite difference weight, and d is the steps size for the sample directions. The finite difference weights are computed optimally [8] and scaled such that the resulting vector from the summation is normalized. The normalization is done so the particles are moved up to the current collision depth, $C_c(\mathbf{p}^i)$, and not in collision free space. The collision constraint is iteratively solved and applied to the particles as shown in lines 5 and 6 in Algorithm 1.

The second constraint, density, ensures that the liquid is in-compressible. The density of particle based representations for liquids can be expressed using the same technique as Smoothed Particle Hydrodynamics (SPH) [11, 24]. SPH simulations compute physical properties from hydrodynamics, such as density, using interpolation techniques with kernel operators centered about the particle locations.

Similarly, we compute the density at particle $\mathbf{p}^i$

$$\rho^i(\mathbf{p}) = \sum_{j=1}^{N} W(||\mathbf{p}^i - \mathbf{p}^j||, h) \tag{4}$$

where $W(\cdot, h)$ is a smoothing kernel operator with radius h. This is the same as SPH simulations except without the mass term because each particle is set to represent an equal amount of mass in the reconstructed liquid. A density constraint for the i-th particle using (5) can be written as:

$$C_\rho^i(\mathbf{p}) = \frac{\rho_i(\mathbf{p})}{\rho_0} - 1 \tag{5}$$

where ρ_0 is the resting density of the liquid being reconstructed [3]. This density constraint is satisfied at 0 which occurs when the reconstructed liquid is achieves resting density at each of the particle locations.

Newton steps along the constraint's gradient are iteratively taken to satisfy the density constraint in (5). Each Newton step, $\Delta\mathbf{p}_\rho$, is calculated as:

$$\Delta\mathbf{p}_\rho = -\nabla\mathbf{C}_\rho(\mathbf{p}) \left(\nabla\mathbf{C}_\rho^\top(\mathbf{p})\nabla\mathbf{C}_\rho(\mathbf{p}) + \epsilon_\rho\mathbf{I}\right)^{-1} \mathbf{C}_\rho(\mathbf{p}) \tag{6}$$

where $\mathbf{C}_\rho(\cdot) = [C_\rho^1(\cdot), \ldots, C_\rho^N(\cdot)]^\top$, the partials are

$$\begin{aligned} \frac{\partial C_\rho^i(\mathbf{p})}{\partial \mathbf{p}^i} &= \frac{1}{\rho_0} \sum_{j=1}^{N} \frac{(\mathbf{p}^i - \mathbf{p}^j)}{||\mathbf{p}^i - \mathbf{p}^j||} W'(||\mathbf{p}^i - \mathbf{p}^j||, h) \\ \frac{\partial C_\rho^i(\mathbf{p})}{\partial \mathbf{p}^j} &= \frac{(\mathbf{p}^i - \mathbf{p}^j)}{\rho_0||\mathbf{p}^i - \mathbf{p}^j||} W'(||\mathbf{p}^i - \mathbf{p}^j||, h) \end{aligned} \tag{7}$$

where $W'(\cdot, h)$ is the derivative of smoothing kernel operator in (4), and $\epsilon_\rho\mathbf{I} \in \mathbb{R}^{N\times N}$ stabilizes the inversion with a damping factor ϵ_ρ. Enforcing incompressibility in SPH simulations, similar to the proposed density constraint here, when particles have a small number of neighbors is known to cause particle clustering [26]. Therefore, we use Monaghan's solution by adding the following artificial pressure term to $\Delta\mathbf{p}_\rho$:

$$\mathbf{s}_{corr}^i = -\frac{\lambda_s}{\rho_0} \sum_{j=1}^{N} \left(\frac{W(||\mathbf{p}^i - \mathbf{p}^j||, h)}{W(\lambda_\mathbf{p}, h)}\right)^{\lambda_n} \frac{\partial C_\rho^i(\mathbf{p})}{\partial \mathbf{p}^j} \tag{8}$$

for the i-th particle where $\lambda_s, \lambda_\mathbf{p}, \lambda_n$ are set according to the original work [26]. The density constraint is iteratively solved with the artificial pressure term and applied to the particles as shown in lines 7 and 8 in Algorithm 1.

3.2. Differentiable Liquid Surface Rendering

The loss being minimized in (1) to reconstruct the liquid is between the detected surface, $\mathbb{I}$, and the reconstructed surface, $\hat{\mathbb{I}}(\cdot)$. This is equivalent to the differentiable rendering problem formulation, which multiple solutions have been proposed for [18]. The differentiable renderer we employ is Pulsar which renders each particle as a sphere [22] because it is currently state-of-the-art for point-based geometry rendering and requires no training data to get a gradient of the rendered image with respect to the particle locations when not using its shader. The loss used to minimize the difference between the detected surface and rendered surface is the Symmetric Mean Absolute Percentage Error (SMAPE):

$$\mathcal{L}\left(\mathbb{I}, \hat{\mathbb{I}}(\mathbf{p})\right) = \frac{1}{N_p} \sum_{u,v\in\mathbb{I}} \frac{|\mathbb{I}_{u,v} - \hat{\mathbb{I}}_{u,v}(\mathbf{p})|}{|\mathbb{I}_{u,v}| + |\hat{\mathbb{I}}_{u,v}(\mathbf{p})| + \epsilon_s} \tag{9}$$

where N_p is the number of pixels on the image and ϵ_s is used to stabilize the division. SMAPE was chosen because the ℓ-1 loss was used in the original Pulsar work [22] and SMAPE is a symmetric version of an ℓ-1 loss. In Algorithm 1, lines 10 and 11 are where the differentiable renderer is integrated into our reconstruction technique with a gradient step size of $\alpha_\mathbb{I}$.

3.3. Adding and Removal of Particles

The number of particles N must be found to solve (1), hence making this a mixed-integer optimization problem. To solve for N, we use a branching strategy based on the following heuristic: if the rendered surface area is smaller than the detected surface area, duplicate a particle, $N + 1$, and vice-versa to remove a particle, $N - 1$. The branching strategy is enabled after confirming a local-minima has been reached with the current number of particles. This is determined by taking the mean image loss gradient and checking if it less than a threshold:

$$\frac{1}{N} \sum_{k=1}^{N} \left|\left| \frac{\partial\mathcal{L}\left(\mathbb{I}, \hat{\mathbb{I}}(\mathbf{p})\right)}{\partial\mathbf{p}^k} \right|\right| \leq \gamma_s \tag{10}$$

where γ_s is the threshold and if the Intersection over Union (IoU) is less than a threshold:

$$\frac{\mathbb{I} \cup \hat{\mathbb{I}}(\mathbf{p})}{\mathbb{I} \cap \hat{\mathbb{I}}(\mathbf{p})} \leq \gamma_I \tag{11}$$

where γ_I is the threshold. IoU is chosen over the SMAPE loss because Pulsar renders each sphere with a blending value so the rendered image will have values from $[0, 1]$ hence increasing the SMAPE loss as more spheres are rendered even when the spheres make a perfect silhouette fit. Meanwhile IoU directly measures silhouette fit which is in line with our heuristic for the branching strategy. If these two criteria are satisfied, a local-minima due to the number of particles is assumed, and the branching decision of duplicating or removing a particle is triggered. This branching logic is handled in lines 12 and 13 in Algorithm 1.

When duplicating or removing a particle, the collision constraint will remain unchanged and the density constraint will be increased when duplicating a particle and decreased when removing a particle. Therefore, the particle selected to duplicate or remove is chosen to best satisfy the density constraint so the initial particle locations when solving (1) after adjusting the particle count remains closest to the density constraint manifold. Written explicitly and using the ℓ-1 loss to describe closeness to the constraint manifold, the index of the particle to duplicate or remove is found by solving

$$\arg\min_i \sum_{k=1}^{N} |C_\rho^{k+}(\mathbf{p})| + |C_\rho^{i+}(\mathbf{p})| \quad \arg\min_i \sum_{k\neq i}^{N} |C_\rho^{k-}(\mathbf{p})| \tag{12}$$

for duplication and removal respectively and $C_\rho^{k+}(\cdot)$, $C_\rho^{k-}(\cdot)$ are the density constraint evaluated at particle k after duplicating and removing the i-th particle respectively. The new density constraints are evaluated as:

$$C_\rho^{k+}(\mathbf{p}) = \frac{1}{\rho_0} \sum_{j=1}^{N+1} W(||\mathbf{p}^k - \mathbf{p}^j||, h) - 1 \tag{13}$$

$$C_\rho^{k+}(\mathbf{p}) = C_\rho^{k}(\mathbf{p}) + \frac{1}{\rho_0} W(||\mathbf{p}^k - \mathbf{p}^{N+1}||, h) \tag{14}$$

for duplicating the i-th particle (so $\mathbf{p}^{N+1} = \mathbf{p}^i$) and

$$C_\rho^{k-}(\mathbf{p}) = \frac{1}{\rho_0} \sum_{j\neq i}^{N} W(||\mathbf{p}^k - \mathbf{p}^j||, h) - 1 \tag{15}$$

$$C_\rho^{k-}(\mathbf{p}) = C_\rho^{k}(\mathbf{p}) - \frac{1}{\rho_0} W(||\mathbf{p}^k - \mathbf{p}^i||, h) \tag{16}$$

for removing the i-th particle where $C_\rho^k(\mathbf{p})$ is the density constraint evaluated at particle k before duplicating or removing a particle. Finally, (12) is solved by explicitly computing the loss for every potential i (i.e. computing loss after duplicating or removing every particle) and choosing i that yields the smallest loss, hence duplicating or removing particles that best satisfy the density constraint. Note that this can be efficiently computed due to the simplifications derived in (14) and (16).

3.4. Liquid Prediction

In Position Based Fluid simulations, the constraints at every time step update the positions of the particles which in turn induces a velocity for the particles [25]. These constraint induced velocities combined with other external forces such as gravity are used to move the particles forward in time for liquid-like motion of the particles. A similar approach is used here to recreate the liquid-like motion through time and hence enable reconstruction from a video of observations, $\mathbb{I}_1 \ldots, \mathbb{I}_T$. Let $\mathbf{p}_t^{i*}$ and $\mathbf{p}_{t-1}^{i*}$ be the optimized particles from solving (1) at time t and $t-1$ respectively. Then the induced velocity for time t is:

$$\mathbf{v}_t^i = (1 - \lambda_d)(\mathbf{p}_t^{i*} - \mathbf{p}_{t-1}^{i*})/\Delta t \tag{17}$$

where $\lambda_d \in [0, 1]$ is the velocity dampening factor and Δt is the time-step size. For consistent motion, XSPH viscosity [38] is applied:

$$\bar{\mathbf{v}}_t^i = \mathbf{v}_t^i + \lambda_v \sum_{j=1}^{N} \frac{\mathbf{v}_t^j - \mathbf{v}_t^i}{\rho^j(\mathbf{p})} W(||\mathbf{p}_t^{i*} - \mathbf{p}_t^{j*}||, h) \tag{18}$$

where λ_v dictates the amount of viscosity applied. The induced velocity is computed after every timestep of liquid reconstruction as shown in lines 14 and 15 in Algorithm 1. The induced velocity and gravity are used to forward predicts the particles to $t+1$ using equations of motion:

$$\mathbf{p}_{t+1}^i = \mathbf{p}_t^{i*} + \bar{\mathbf{v}}_t^i \Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 \tag{19}$$

where $\mathbf{g}$ is the gravity vector. This forward prediction is done in line 1 of Algorithm 1. The dampening and viscosity not only represent physical properties, but also provide tuning parameters to stabilize the initialization for the next timestep. Dampening, λ_d, dictates how much to rely on the prediction and viscosity, λ_v, adjusts the consistency of the velocity.

4. Experiments

To show the effectiveness of our proposed liquid reconstruction method, we test on a simulated and two real-life datasets with a comparative study. These experiments are explained in the coming sections, and first implementation details are given. Secondly, a description of our datasets and how they are collected is presented. Lastly, we explain our comparative study and the results of it on the datasets.

4.1. Implementation Details

All the arithmetic, e.g. Newton's density constraint step in (6), are implemented with PyTorch for its GPU integration [32]. The collision constraint, (2), and its solution, (3), are implemented with SPNet's *ConvSP* operator and its PyTorch wrapper [41], Kernel $K = \{[\pm 1, 0, 0], [0, \pm 1, 0], [0, 0, \pm 1]\}$, and step size d is equal to the resolution of the $SDF(\cdot)$. The resting density ρ_0 is generated by setting a resting distance between particles because that is more intuitive to adjust. The resting distance between particles is converted to the resting density by packing 1000 particles in a sphere and computing the particle density of the sphere. Then the density constraint parameters to solve (5) and (6) are set to a resting distance of $0.6h$, $\epsilon_p = 10^2$, and $W(\cdot)$ is set to Poly6 and Spiky

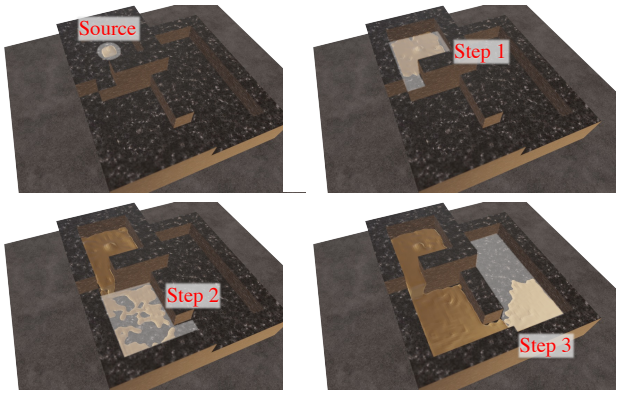

Figure 3. The sequence of figures (top-left, top-right, bottom-left, bottom-right) shows how liquid fills the simulated fountain. Take note how the first step fills in a consistent shape, but significant turbulence occurs when dropping to the second step making this a challenging component of the scene. Another challenge is by the time the third and final step of the fountain fills, a significant number of particles must be used for reconstruction due to the large volume, hence testing the scalability of the reconstruction method.

Kernels for density estimation and gradient steps respectively [28]. Differentiable rendering is done with the PyTorch3D framework [36] and $\epsilon_s = 10^{-2}$. The thresholds for adding/removing particles are $\gamma_s = 10^{-3}$ and $\gamma_I = 0.9$ respectively. Velocity dampening and viscosity coefficients are set to $\lambda_d = 0.2$ and $\lambda_v = 0.75$ respectively. The parameters in Algorithm (1) are set to $n_o = 30$, $n_j = 2$, $n_c = 5$, $n_i = 5$, and $\alpha_{\mathbb{I}} = 0.02$. All datasets are stereo so an initial four particles can be placed at a stereo-computed, 3D location from the first liquid detections. The last parameters, $SDF(\cdot)$ resolution and particle interaction radius, h, are set depending on the dataset as they need to be adjusted depending on the scale of the reconstruction.

4.2. Datasets

Simulated Fountain: The first dataset is generated on a three step fountain, shown in Fig. 3, with Blender [5]. The liquid simulation uses all default values except the viscosity is set to 0.001. The $SDF(\cdot)$ is generated from the fountain with a resolution of 1cm and the particle interaction radius, h, is set to 1cm. The scene is rendered with 1080p at 24fps stereo cameras, and a mask of the rendered liquid is directly outputted from Blender. For this simulated dataset, the ground-truth liquid mesh is available to evaluate our reconstruction with. The metric of 3D IoU is used to capture the shape accuracy of our reconstruction and computed as:

$$\text{IoU}_{3D} = \frac{\mathbb{V} \cup \hat{\mathbb{V}}(\mathbf{p})}{\mathbb{V} \cap \hat{\mathbb{V}}(\mathbf{p})} \tag{20}$$

where $\mathbb{V}$ and $\hat{\mathbb{V}}(\mathbf{p})$ are voxel representation of the simulated and reconstructed liquid respectively. The reconstructed liquid in voxel representation, $\hat{\mathbb{V}}(\mathbf{p})$, is generated with the color field, shown in equation (21) in the supplementary material. The voxel grid is computed at a resolution of 3cm.

Figure 4. The left figure shows a top down view of the silicon cavity used for the Endoscopic Liquid dataset, and the liquid is injected with a syringe at the labelled points for three trials. The right figure shows a camera image from our Pouring Milk experiment set up. Notice that the milk is partially blocked by the mug, hence testing the reconstructions ability to handle occlusions.

Endoscopic Liquid: The second dataset uses a custom silicon cavity that was molded with a 3D printed negative so a $SDF(\cdot)$ for it can be generated. The cavity is 11cm by 9.5cm, $SDF(\cdot)$ resolution is set to 1mm, and particle interaction radius, h, is set to 5mm. To transform the $SDF(\cdot)$ to the camera frame, which is the coordinate frame the particles are being optimized in, an ArUco Marker [10] is placed on the cavity in a known location. Roughly 50ml of water is injected with a syringe at three different locations for three trails as depicted in Fig. 4. The water is mixed with red-coloring dye so color segmentation can be applied to detect the liquid surface. The liquid video is recorded using a da Vinci Research Kit stereo-endoscope which is 1080p at 30fps [19].

Pouring Milk: The third dataset is pouring chocolate milk by a human into a mug as shown in Fig. 4. The mug is 9cm high and has a 7cm diameter, the $SDF(\cdot)$ resolution is set to 1mm, and particle interaction radius, h, is set to 6.5mm. The mug is placed on a sheet of paper with an ArUco Marker [10] in a marked location. The Aruco Marker and known geometry of the paper provides the transformation to take the $SDF(\cdot)$ to the camera frame. Color segmentation is used to detect the chocolate milk's liquid surface. The liquid video is recorded at 720p 15fps using a ZED Stereo Camera from Stereo Labs.

4.3. Comparative Study

We show the effectiveness of our proposed method through a comparative study. The configurations being compared are:

- *No Constraints* [22] (i.e. no density or collision constraints) and only image loss
- *No Density* constraint
- *No Collision* constraint

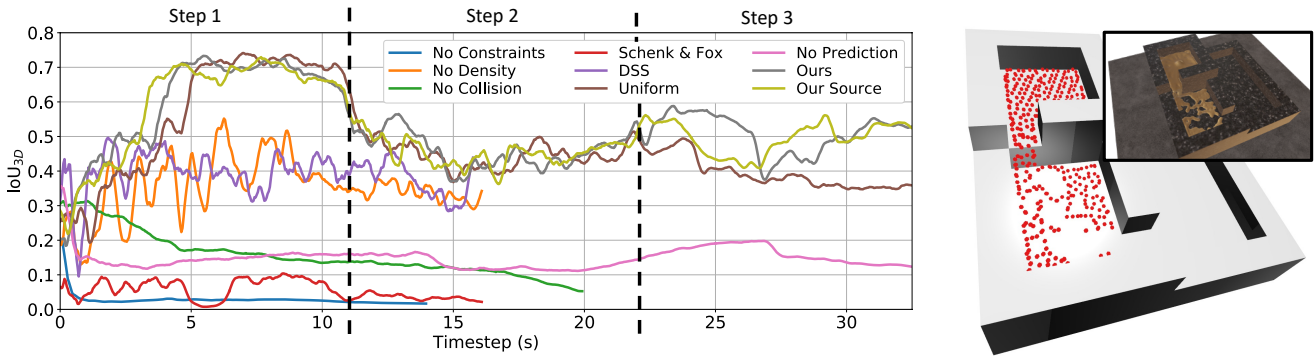

Figure 5. The plot on the left shows IoU_{3D} results from the Simulated Fountain datasets along with time-marked points when the liquid reaches different steps in the scene. Note how our proposed methods and the uniform comparison are able to reach 70% IoU_{3D} in the first step, and retain a good reconstruction as the very long, and turbulent simulation continues. An example of our reconstruction approach during the turbulent period of the scene is shown on the right-hand figure. Meanwhile the compared approaches ran into memory limitations and crashed (required greater than 24GB of memory) or were unable to converge effectively.

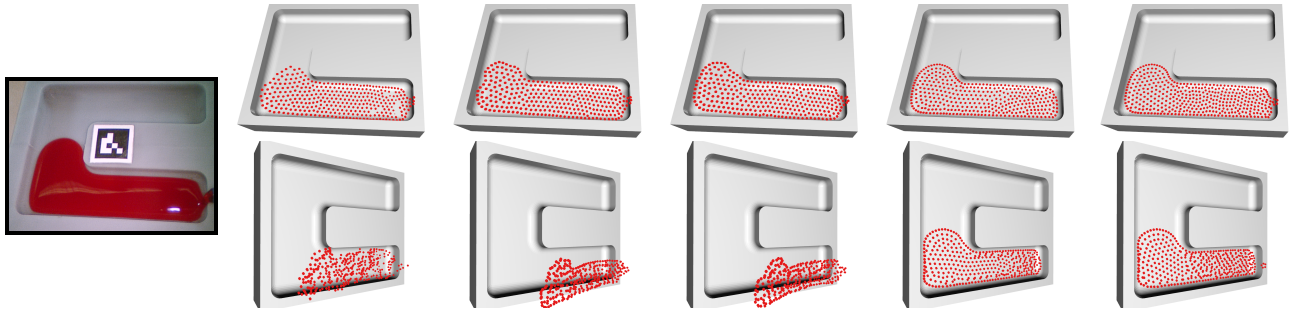

Figure 6. From left to right the image columns are an endoscopic image of liquid being reconstruction with: no constraints, no collision constraint, no prediction, our approach, and our source estimation technique. The first row of renderings have the virtual camera positioned similar to the real endoscope showing how from that perspective, the particles in red line up with the real image of the liquid. The second row shows another rendered perspective and how our proposed approaches properly reconstruct the liquid in 3D. The first three comparisons are unable to properly reconstruct because they do not leverage a liquids dynamics (i.e. falling to gravity and colliding with the cavity).

- *Schenck & Fox* [41] constraints instead of the density constraint we presented
- *DSS* [46] for rendering gradients rather than Pulsar
- *Uniform* random selection for duplication or removal of particles instead of solving (12)
- *No Prediction* of particles (line 1 in Algorithm 1)
- *Our* complete approach
- *Our Source* estimation which adds particles at a source location and detailed in the next sub-section

When removing the collision constraint (i.e. No Constraints and No Collision comparisions), the particle prediction also had to be turned off otherwise the particles will fall forever due to gravity. Without the density or collision constraint, the method is equivalent to differentiable rendering [22] thus giving a baseline comparison. Schenck & Fox proposed their own position-based liquid constraints for constant density [41] (i.e. replacing $\mathbf{C}_\rho$) which were integrated into this method for comparison. We also compared with another recently developed differentiable renderer for point-based geometry called Differentiable Surface Splatting (DSS) [46]. Lastly, a source estimation technique is implemented to highlight how the proposed method can be extended. Implementation details for the Schenk & Fox, DSS, and Our Source comparisons are given the supplementary material.

Videos and convergence statistics of the comparison study on all the datasets are in the supplementary materials, and a few highlights are given here. Quantitative results from the Simulated Fountain dataset are shown in Fig. 5, and it shows how effective our proposed approach is in a turbulent, long scene. Fig. 6 shows results from the Endoscopic Trails and how our proposed methods leverage liquid dynamics to fit the cavity shape correctly. From the Milk Pouring experiments, results are shown in Fig. 7 and 8 which indicate that our proposed method is able to infer liquid in occluded regions and reconstruct the falling stream.

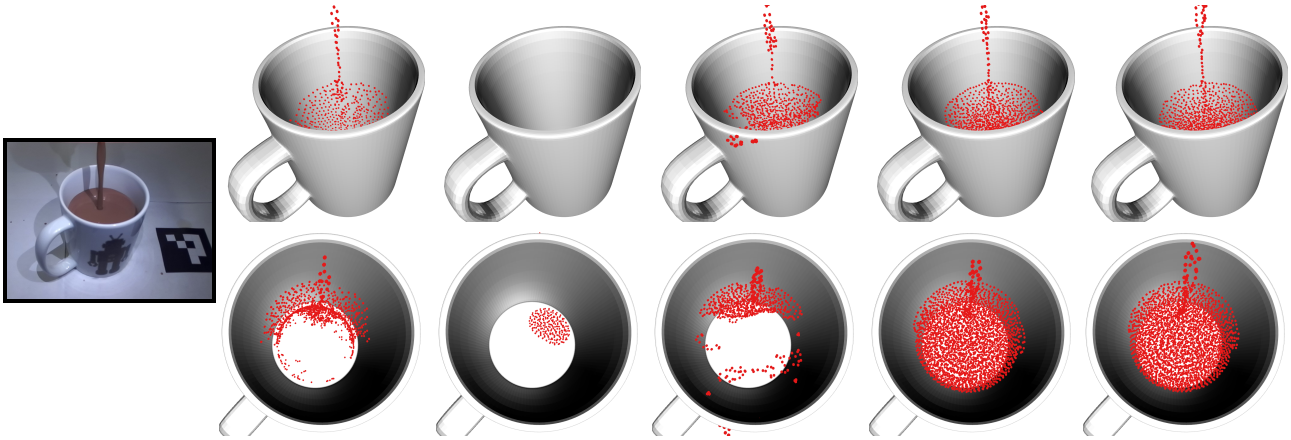

Figure 7. From left to right the image columns are an image from the Pouring Milk dataset being reconstructed with: no density constraint, Schenck & Fox constraints [41], DSS [46], our approach, and our source estimation technique. The first row of renderings have the virtual camera positioned similar to the raw image showing how from that perspective, the particles in red line up with the real image of the liquid. The second row shows a birds-eye-view perspective and how our proposed approaches properly reconstruct the liquid in 3D. The no density constraint and DSS [46] comparisons are unable to properly reconstruct due to over-fitting on the image loss and fail to make inferences in the occluded region. Meanwhile Schenck & Fox constraints [41] constraints went unstable and splashed particles outside the mug.

Figure 8. The left-most image is from the Pouring Milk dataset being reconstructed with the uniform comparison (middle) and our method (right). Our method is able to reconstruct the falling stream, unlike the uniform comparison, due to the novel particle insertion and removal approach.

5. Discussion and Conclusion

Our method is the first approach to reconstructing liquids with only knowledge of the collision environment, gravity direction, and 2D surface detections. We limited the scope to 2D surface detections because a liquids color is too variable from reflections and refractions. Our experiments highlight the generalizability of our approach through the wide range of liquids (simulated, water, and milk) and cameras (narrow & wide field of view and 15, 24 & 30 fps). In the supplementary material video, consistent particle flow is observed when using the source estimation extension. We envision that the source estimation extension will be beneficial in downstream robotic automation applications such as robotic bar tending [44] and managing hemostasis in surgery [37] where prediction of the liquid is required.

We found that the density constraint, collision constraint, and prediction are crucial to inferring beyond the 2D image loss as seen in Fig. 6 and 7. Furthermore in longer and more turbulent scenes, the lack of liquid properties can cause instabilities and blow up the mixed-integer optimizer (greater than 10,000 particles). The density constraint can be switched with other constraints that reflect a liquid incompressibility and other liquid properties, such as Schenck & Fox's constraints [41]. However, we were unable to stabilize Schenck & Fox's constraints and found the constraint in (5) and its solver to be stable on all of our datasets. Similar is true for the differentiable rendering, and we found Pulsar to be more robust than DSS in our application since DSS requires normals which we observed are not consistently generated. Our particle insertion and removal strategy was effective and even able to insert particles to reconstruct a falling stream as seen in Fig. 8.

There is a large quantity of hyper-parameters in our method, but this is expected when solving a mixed-integer, optimization problem. Nevertheless, we found a set that generalizes over our diverse datasets, and the interaction radius, h, adjusts the effective resolution of our reconstruction (i.e. smaller h gives a denser reconstruction). An artifact that we observed in our reconstruction approach is the ambiguity of depth due to our observations being limited to only 2D surface detections of the liquid. The incorporation of liquid dynamics does help overcome this challenge in the Endoscopic Liquid experiment as seen in Fig. 6. However, in the Pouring Milk experiment the top layer of milk is slightly lopsided which is best seen in Fig. 9 in the supplemental materials. In future work, we intend on solving the ambiguity better by modifying (1) to incorporate multiple timesteps, and hence optimizing with the dynamics.

Acknowledgements This work was supported by NSF Career Award #2045803 and US Army TATRC. F. Richter is supported via the NSF Graduate Research Fellowships.

References

[1] Bradley Atcheson, Wolfgang Heidrich, and Ivo Ihrke. An evaluation of optical flow algorithms for background oriented schlieren imaging. *Experiments in fluids*, 46(3):467–476, 2009. 2

[2] Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus Magnor, and Hans-Peter Seidel. Time-resolved 3d capture of non-stationary gas flows. *ACM transactions on graphics (TOG)*, 27(5):1–9, 2008. 2

[3] Kenneth Bodin, Claude Lacoursiere, and Martin Servin. Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):516–526, 2011. 4

[4] Robert Bridson. *Fluid simulation for computer graphics*. CRC press, 2015. 1

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 6

[6] S áB Dalziel, Graham O Hughes, and Bruce R Sutherland. Whole-field density measurements by 'synthetic schlieren'. *Experiments in fluids*, 28(4):322–335, 2000. 2

[7] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019. 2

[8] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988. 3

[9] Erik Franz, Barbara Solenthaler, and Nils Thuerey. Global transport for fluid reconstruction with learned self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1632–1642, 2021. 1, 2

[10] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 6

[11] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977. 3

[12] Ian Grant. Particle image velocimetry: a review. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 211(1):55–76, 1997. 2

[13] Jinwei Gu, Shree K Nayar, Eitan Grinspun, Peter N Belhumeur, and Ravi Ramamoorthi. Compressive structured light for recovering inhomogeneous participating media. *IEEE transactions on pattern analysis and machine intelligence*, 35(3):1–1, 2012. 2

[14] Tatiana López 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 *Proceedings of the Conference on Robot Learning (CoRL)*, 2017. 2

[15] Tim Hawkins, Per Einarsson, and Paul Debevec. Acquisition of time-varying participating media. *ACM Transactions on Graphics (ToG)*, 24(3):812–815, 2005. 2

[16] Jingbin Huang, Fei Liu, Florian Richter, and Michael C Yip. Model-predictive control of blood suction for surgical hemostasis using differentiable fluid simulations. *IEEE International Conference on Robotics and Automation*, 2021. 1

[17] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016. 1

[18] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 4

[19] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. An open-source research kit for the da vinci® surgical system. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 6434–6439. IEEE, 2014. 6

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 12

[21] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013. 1

[22] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021. 4, 6, 7, 13, 14

[23] Yang Li, Florian Richter, Jingpei Lu, Emily K Funk, Ryan K Orosco, Jianke Zhu, and Michael C Yip. Super: A surgical perception framework for endoscopic tissue manipulation with surgical robotics. *IEEE Robotics and Automation Letters*, 5(2):2294–2301, 2020. 1

[24] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977. 3

[25] Miles Macklin and Matthias Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 2, 5

[26] Joseph J Monaghan. Sph without a tensile instability. *Journal of computational physics*, 159(2):290–311, 2000. 4

[27] Roozbeh Mottaghi, Connor Schenck, Dieter Fox, and Ali Farhadi. See the glass half full: Reasoning about liquid containers, their volume and content. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1871–1880, 2017. 2

[28] Matthias Müller, David Charypar, and Markus H Gross. Particle-based fluid simulation for interactive applications. In *Symposium on Computer animation*, pages 154–159, 2003. 6, 11, 12

[29] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. 2

[30] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. 1

[31] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011. 1

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[33] Stephen B Pope. Turbulent flows, 2001. 1

[34] Arturo Rankin and Larry Matthies. Daytime water detection based on color variation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 215–221. IEEE, 2010. 2

[35] Arturo L Rankin, Larry H Matthies, and Paolo Bellutta. Daytime water detection based on sky reflections. In *2011 IEEE International Conference on Robotics and Automation*, pages 5329–5336. IEEE, 2011. 2

[36] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 6

[37] Florian Richter, Shihao Shen, Fei Liu, Jingbin Huang, Emily K Funk, Ryan K Orosco, and Michael C Yip. Autonomous robotic suction to clear the surgical field for hemostasis using image-based blood flow detection. *IEEE Robotics and Automation Letters*, 6(2):1383–1390, 2021. 2, 8

[38] Hagit Schechter and Robert Bridson. Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012. 5

[39] Connor Schenck and Dieter Fox. Visual closed-loop control for pouring liquids. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2629–2636. IEEE, 2017. 2

[40] Connor Schenck and Dieter Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018. 2

[41] Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR, 2018. 2, 5, 7, 8, 11, 13, 14

[42] Nguyen Truong-Thinh, Nguyen Ngoc-Phuong, and Tuong Phuoc-Tho. A study of pipe-cleaning and inspection robot. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2593–2598. IEEE, 2011. 1

[43] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2442–2447. IEEE, 2017. 1

[44] Hongtao Wu and Gregory S Chirikjian. Can i pour into it? robot imagining open containability affordance of previously unseen objects via physical simulations. *IEEE Robotics and Automation Letters*, 6(1):271–278, 2020. 1, 8

[45] Akihiko Yamaguchi and Christopher G Atkeson. Stereo vision of liquid and particle flow for robot pouring. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1173–1180. IEEE, 2016. 2

[46] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 7, 8, 11, 13, 14

[47] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)*, 32(1):1–12, 2013. 11

[48] Guangming Zang, Ramzi Idoughi, Congli Wang, Anthony Bennett, Jianguo Du, Scott Skeen, William L Roberts, Peter Wonka, and Wolfgang Heidrich. Tomofluid: reconstructing dynamic fluid from sparse view videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1870–1879, 2020. 2

[49] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing, 2018. 12