Detection and Tracking of Liquids with Fully Convolutional Networks

Connor Schenck, Dieter Fox
University of Washington
{schenckc,fox}@cs.washington.edu

***Abstract*—Recent advances in AI and robotics have claimed many incredible results with deep learning, yet no work to date has applied deep learning to the problem of liquid perception and reasoning. In this paper, we apply fully-convolutional deep neural networks to the tasks of detecting and tracking liquids. We evaluate three models: a single-frame network, multi-frame network, and a LSTM recurrent network. Our results show that the best liquid detection results are achieved when aggregating data over multiple frames, in contrast to standard image segmentation. They also show that the LSTM network outperforms the other two in both tasks. This suggests that LSTM-based neural networks have the potential to be a key component for enabling robots to handle liquids using robust, closed-loop controllers.**

I. Introduction

To robustly handle liquids, such as pouring a certain amount of water into a bowl, a robot must be able to perceive and reason about liquids in a way that allows for closed-loop control. Liquids present many challenges compared to solid objects. For example, liquids can not be interacted with directly by a robot, instead the robot must use a tool or container; often containers containing some amount of liquid are opaque, obstructing the robot's view of the liquid and forcing it to remember the liquid in the container, rather than re-perceiving it at each timestep; and finally liquids are frequently transparent, making simply distinguishing them from the background a difficult task. Taken together, these challenges make perceiving and manipulating liquids highly non-trivial.

Recent advances in deep learning have enabled a leap in performance not only on visual recognition tasks, but also in areas ranging from playing Atari games [5] to end-to-end policy training in robotics [12]. In this paper, we investigate how deep learning techniques can be used for perceiving liquids during pouring tasks. We develop a method for generating large amounts of labeled pouring data for training and testing using a realistic liquid simulation and rendering engine, which we use to generate a data set with over 4.5 million labeled images. Using this dataset, we evaluate multiple deep learning network architectures on the tasks of detecting liquid in an image and tracking the location of liquid even when occluded.

The rest of this paper is laid out as follows. Section II describes work related to ours. Section III details our experimental methodology. Section IV describes how we evaluated the neural networks. Section V details our results. Section VI contains a discussion of the implications of the results and the conclusions that can be drawn from them. And finally section VII details our directions for future work.

II. Related Work

To the best of our knowledge, no prior work has investigated directly perceiving and reasoning about liquids. Existing work relating to liquids either uses coarse simulations that are disconnected to real liquid perception and dynamics [10, 21] or constrained task spaces that bypass the need to perceive or reason directly about liquids [11, 15, 20, 2, 19]. While some of this work has dealt with pouring, none of it has attempted to directly perceive the liquids from raw sensory data, in contrast to this paper, in which we investigate ways to do just that.

Although there has been some prior work in robotics that has dealt with perception and liquids, though in constrained task spaces. Work by Rankin *et al.* [16, 17] investigated ways to detect pools of water for an unmanned ground vehicle navigating rough terrain. However they detected water based on simple color features or sky reflections, and didn't reason about the dynamics of the water, instead treating it as a static obstacle. Griffith *et al.* [4] used the auditory and proprioceptive feedback from objects interacted with in a sink environment with a running water tap in order to learn about those objects, although in this case the robot did not detect or reason about the water, rather it used the water as a means to learn about and categorize other objects. In contrast to [4], we use vision to directly detect the liquid itself, and unlike [16, 17], we treat the liquid as dynamic and reason about it, rather than treating it as a static obstacle.

In order to perceive liquids at the pixel level, we make use of fully-convolutional neural networks (FCN). FCNs have been successfully applied to the task of image segmentation in the past [13, 6, 18] and are a natural fit for pixel-wise classification. In addition to FCNs, we utilize long short-term memory (LSTM) [7] recurrent cells to reason about the temporal evolution of liquids. LSTMs are preferable over more standard recurrent networks for long-term memory as they overcome many of the numerical issues during training such as exploding gradients [3]. LSTM-based CNNs have been successfully applied to many temporal memory tasks by previous work [14, 18], and in fact [18] even combine LSTMs and FCNs by replacing the standard fully-connected layers of their LSTMs with 1×1 convolution layers. We use a similar method in this paper.

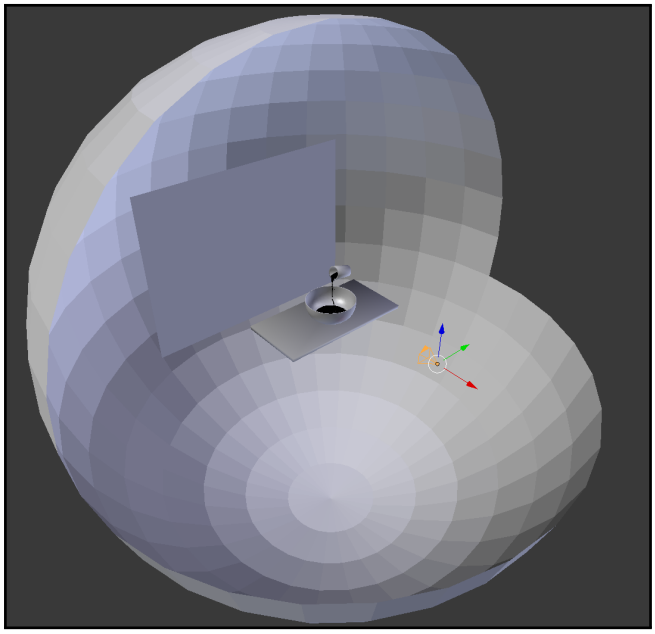


Fig. 1: The setup used to simulate and render liquid sequences. The objects are shown here textureless for clarity. The sphere surrounding all the objects has been cut away to allow viewing of the objects inside. The orange shape represents the camera's viewpoint, and the flat plane across the table from it is the plane on which the video sequence is rendered. Note that this plane is sized to exactly fill the camera's view frustum. The background sphere is not directly visible by the camera and is used primarily to compute realistic reflections.

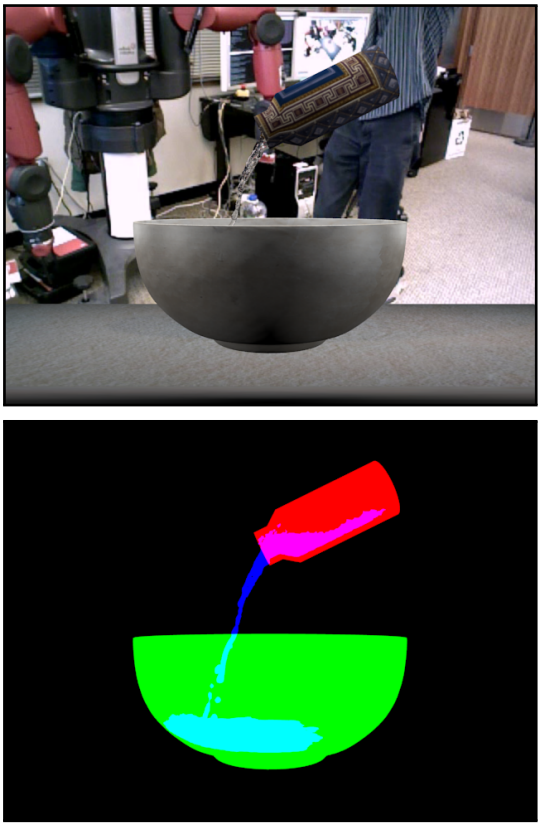


Fig. 2: An example of a frame rendered by our data generation algorithm. The upper image is the raw RGB image generated by the renderer. The lower image is the ground truth binary pixel labels, where the blue channel labels the liquid pixels, the green channel the bowl, and the red channel the cup. The alpha channel (not shown) indicates which of the three (liquid, bowl, cup), if any, is visible at that pixel.

III. Methodology

In order to train neural networks to perceive and reason about liquids, we must first have labeled data to train on. Getting pixel-wise labels for real-world data can be difficult, so in this paper we opt to use a realistic liquid simulator. In this way we can acquire ground truth pixel labels while generating images that appear as realistic as possible. We train three different types of convolutional neural networks (CNNs) on this generated data to detect and track the liquid: single-frame CNN, multi-frame CNN, and LSTM-CNN.

A. Data Generation

We generate data using the 3D-modeling application Blender [1] and the library El'Beem for liquid simulation, which is based on the lattice-Boltzmann method for efficient, physically accurate liquid simulations [9]. We separate the data generation process into two steps: simulation and rendering. During simulation, the liquid simulator calculates the trajectory of the surface mesh of the liquid as the cup pours the liquid into the bowl. We vary 4 variables during simulation: the type of cup (cup, bottle, mug), the type of bowl (bowl, dog dish, fruit bowl), the initial amount of liquid (30% full, 60% full, 90% full), and the pouring trajectory (slow, fast, partial), for a total of 81 simulations. Each simulation lasts exactly 15 seconds for a total of 450 frames (30 frames per second).

Next we render each simulation. We separate simulation from rendering because it allows us to vary other variables that don't affect the trajectory of the liquid mesh (e.g., camera viewpoint), which provides a significant speedup as liquid simulation is much more computationally intensive than rendering. In order to approximate realistic reflections, we mapped a 3D photo sphere image taken in our lab to the inside of a sphere, which we placed in the scene surrounding all the objects. To prevent overfitting to a static background, we also add a plane in the image in front of the camera and behind the objects that plays a video of activity in our lab that approximately matches with that location in the background sphere. This setup is shown in figure 1. The liquid is always rendered as 100% transparent, with only reflections, refractions, and specularities differentiating it from the background. For each simulation, we vary 6 variables: camera viewpoint (48 preset viewpoints), background video (8 videos), cup and bowl textures (6 textures each), liquid reflectivity (normal, none), and liquid index-of-refraction (air-like, low-water, normal-water). Additionally, we also generate negative examples without liquid. In total, this yields 165,888 possible renders for each simulation. It is infeasible to render

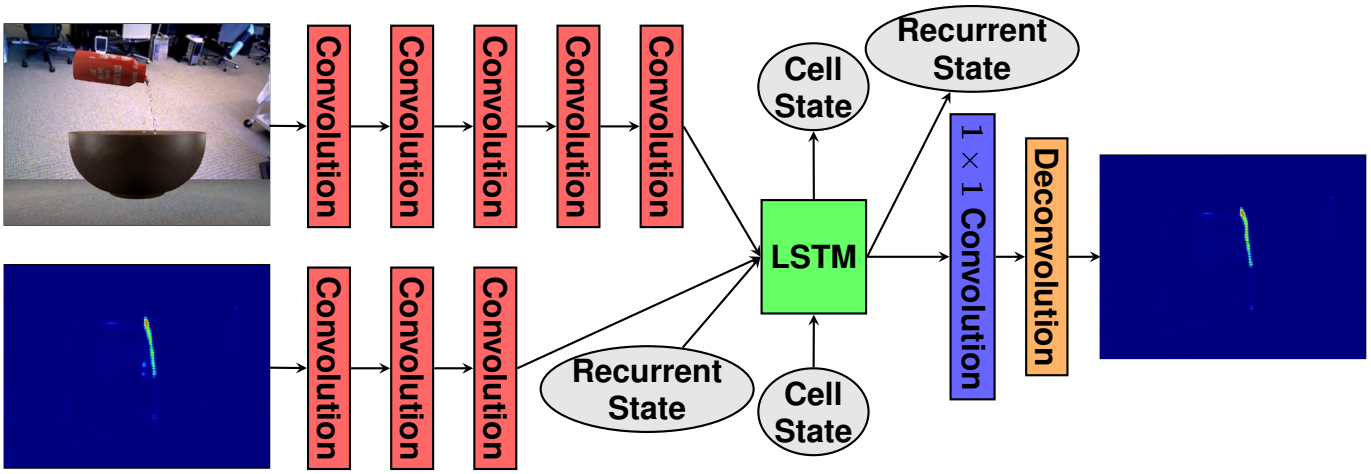


Fig. 3: Layout of the LSTM-CNN. It takes as input the current frame as well as its own predictions from the previous timestep. During training we initialize this at the first timestep as ground truth, but during testing we initialize it as all zeros. The LSTM takes as recurrent input its own output from the previous timestep and the cell state. Refer to figure 1 of [3] for more details. Each of the convolution layers is followed by rectified linear and max pooling layers. The 1×1 convolution layer is followed by a rectified linear layer.

them all, so we randomly sample variable values to render.

The labels are generated for each object (liquid, cup, bowl) as follows. First, all other objects in the scene are set to render as invisible. Next, the material for the object is set to render as a specific, solid color, ignoring lighting. The sequence is then rendered, yielding a class label for the object for each pixel. An example of labeled data and its corresponding rendered image is shown in figure 2. The cup, bowl, and liquid are rendered as red, green and blue respectively. Note that this method allows each pixel to have multiple labels, e.g., some of the pixels in the cup are labeled as both cup and liquid (magenta in the lower part of figure 2). To determine which of the objects, if any, is visible at each pixel, we render the sequence once more with all objects set to render as their respective colors, and we use the alpha channel in the ground truth images to encode the visible class label.

To evaluate our learning architectures, we generated 10,122 pouring sequences by randomly selecting render variables as described above as well as generating negative sequences (i.e., sequences without any water), for a total of 4,554,900 training images. For simplicity, we only used sequences rendered from 6 of the 48 possible camera poses.

B. Detecting and Tracking liquids

We test three network layouts for the tasks of detecting and tracking liquids: CNN, MF-CNN, and LSTM-CNN.

- **CNN** The first layout is a standard convolutional neural network (CNN) with a fixed number of convolutional layers, each followed by a rectified linear layer and a max pooling layer. In place of fully-connected layers, we use two 1×1 convolutional layers, each followed by a rectified linear layer. The last layer of the network is a deconvolutional layer.
- **MF-CNN** The second layout is a multi-frame CNN. Instead of taking in a single frame, it takes as input multiple consecutive frames and predicts at the last frame. Each frame is convolved independently through the first part of the network, which is composed of a fixed number of convolutional layers, each followed by a rectified linear and max pooling layer. The output for each frame is then concatenated together channel-wise, and then fed to two 1×1 convolutional layers, each followed by a rectified linear layer, and finally a deconvolutional layer. We fix the number of input frames for this layout to 32 for this paper, i.e., approximately 1 second's worth of data (30 frames per second).
- **LSTM-CNN** The third layout is similar to the single frame CNN layout, with the first 1×1 convolutional layer replaced with a LSTM layer (see figure 1 of [3] for a detailed layout of the LSTM layer). We replace the fully-connected layers of a standard LSTM with 1×1 convolutional layers. The LSTM takes as recurrent input the cell state from the previous timestep, its output from the previous timestep, and the output of the network from the previous timestep processed through 3 convolutional layers (each followed by a rectified linear and max pooling layer). During training, when unrolling the LSTM CNN, we initialize this last recurrent input with the ground truth at the first timestep, but during testing we initialize it with all zeros.

Figure 3 shows the layout of the LSTM-CNN. For the tracking task, we reduce the number of initial convolution layers on the input from 5 to 3 for each network.

We use the Caffe deep learning framework [8] to implement our networks.

IV. Evaluation

We evaluated the three network types on both the detection and tracking tasks. For detection, the networks were given the full rendered RGB image as input (similar to the top image

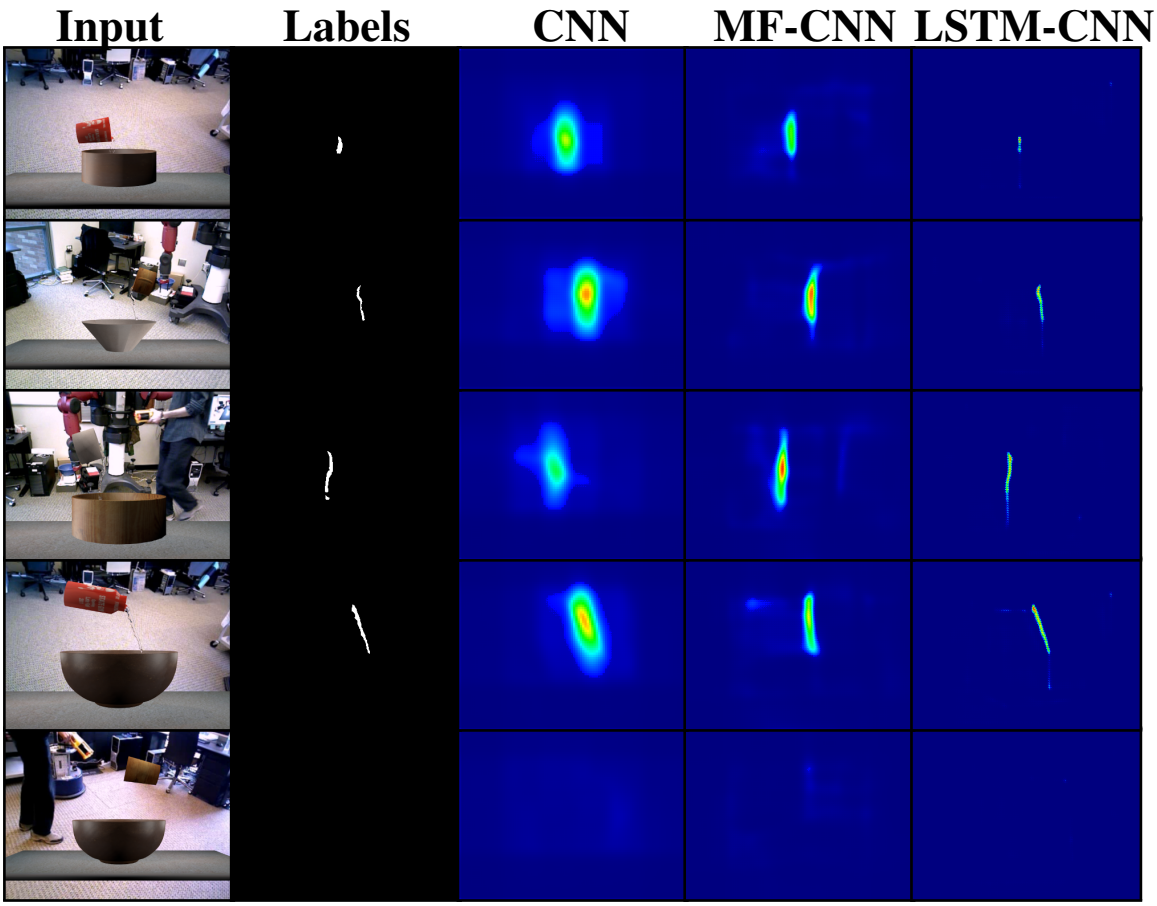


Fig. 4: Qualitative liquid detection results. The Input column is the input to the networks, the Labels column is the ground truth labeling of each pixel as liquid or not liquid, and the CNN, MF-CNN, and LSTM-CNN columns show a heatmap of the prediction of each network for each of the input frames. 5 sequences were randomly selected from our training set, and the frame with the most liquid pixels was picked for display here, with the exception of the last row, which shows how the networks perform when there is no liquid present.

in figure 2) at a resolution of 400×300 pixels. The output was a classification at each pixel as liquid or not liquid. Each network was first trained for 60,000 iterations on image crops of visible liquid, and then again for another 60,000 iterations on the full image (the use of only convolutional layers rather than fully connected layers allows for variable sized inputs and outputs). The weights of the LSTM-CNN were initialized with the weights of the single-frame CNN trained on only cropped image patches. During training, the LSTM-CNN was unrolled for 32 timesteps.

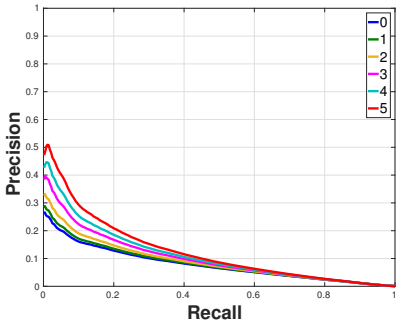
For tracking, the networks were given pre-segmented input images, with the goal being to track the liquid when it is not visible. The input was similar to the bottom image from figure 2, except that only visible liquid was shown (in the case of figure 2, the cyan and magenta liquid would not have been shown because it was occluded by the bowl and cup respectively). Because these input images are more structured, we lowered the resolution to 130×100. The output was the pixel-wise classification of liquid or not liquid, including pixels where the liquid was occluded by other objects in the scene. During training, the LSTM-CNN was unrolled for 180 timesteps.

V. Results

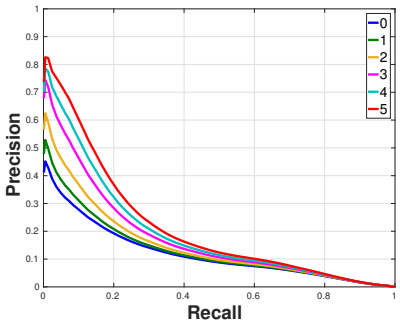
A. Detection Results

Figure 4 shows qualitative results for the three networks on the liquid detection task. The sequences used for this figure were randomly selected from the training set, and the frame with the most liquid visible was selected for display here[1]. It is clear from the figure that all three networks detect the liquid at least to some degree. The single frame CNN is less accurate at a pixel level, but it is still able to broadly detect the presence and general vicinity of liquid. As expected, the multi-frame CNN is much more precise than the single frame CNN. Surprisingly, the LSTM CNN output appears much more accurate than even the multi-frame CNN.

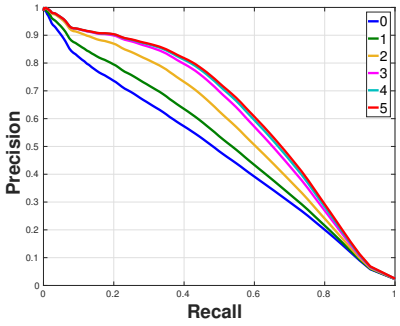
Figure 5 shows a quantitative comparison between the three networks. It plots the precision-recall curves for each of the networks when classifying each pixel as liquid. We plot multiple lines for different amounts of "slack," i.e., how many pixels a positive classification is allowed to be from a positive ground truth pixel and still count as correct. We add this analysis due to the outputs shown in figure 4; the networks



(a) CNN

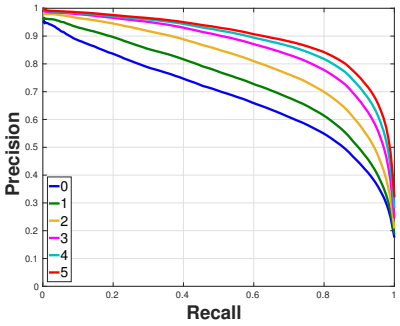


(b) MF-CNN

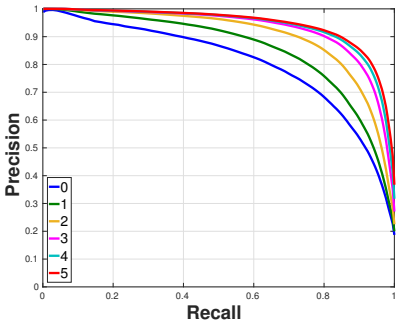


(c) LSTM-CNN

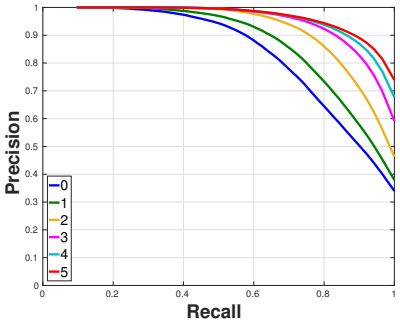
Fig. 5: Quantitative liquid detection results. The graphs indicate the precision and recall for each of the three networks. The colored lines indicate the variation in the number of slack pixels we allowed for prediction, i.e., how many pixels a positive classification could be away from a positive ground truth labeling and still be counted as correct.



(a) CNN



(b) MF-CNN



(c) LSTM-CNN

Fig. 6: Quantitative liquid tracking results. Similar to figure 5, the graphs indicate the precision and recall for each of the three networks and the colored lines indicate the variation in the number of slack pixels we allowed for prediction.

clearly are able to detect the liquid, but are not necessarily pixel-for-pixel perfect. However, for the purposes of liquid manipulation, perfect pixel-wise accuracy is not necessary.

The quantitative results in figure 5 confirm the qualitative outputs shown in figure 4. As expected, the multi-frame CNN outperforms the single-frame. Surprisingly, the LSTM CNN performs much better than both by a significant margin, and even gets a significant boost from only a few slack pixels, indicating that even if the LSTM CNN is not always pixel-for-pixel accurate, it is often very close. Taken together, these results strongly suggest that detecting transparent liquid must be done over a series of frames, rather than a single frame.

B. Tracking Results

For tracking, we evaluated the performance of the networks on locating both visible and invisible liquid, given segmented input (i.e., each pixel classified as liquid, cup, bowl, or background). Because the viewpoint was fixed level with the bowl, the only visible liquid the network was given was liquid as it passed from cup to bowl. Figure 6 shows the performance

of each of the three networks, and the accompanying video[1] shows the tracking results from the same 5 sequences shown in figure 4. Once again we plot the precision-recall curves for each network for different thresholds for the amount of "slack" given to each positive classification (i.e., the number of pixels a positive classification is allowed to be from a true positive pixel to count as correct). As expected, the LSTM CNN has the best performance since it is the only network that has a memory, which is necessary to keep track of the occluded water in the cup and bowl. Interestingly, the multi-frame CNN performs better than expected, given that it only sees approximately 1 second's worth of data and has no memory capability. We suspect this is due to the network's ability to infer the likely location of the liquid based on the angle of the cup and the direction it's moving. The video further reinforces this, as it is clear that in the sequence without liquid (the final sequence), the multi-frame CNN incorrectly infers the existence of water, whereas the LSTM CNN does not.

VI. Discussion & Conclusion

The results in section V show that it is possible for deep learning to independently detect and track liquids in a scene. Unlike prior work on image segmentation, these results clearly show that single images are not sufficient to reliably perceive liquids. Intuitively, this makes sense, as a transparent liquid can only be perceived through its refractions, reflections, and specularities, which vary significantly from frame to frame, thus necessitating aggregating information over multiple frames. We also found that LSTM-based CNNs are best suited to not only aggregate this information, but also to track the liquid as it moves between containers. LSTMs work best, due to not only their ability to perform short term data integration (just like the MF-CNN), but also to remember states, which is crucial for tracking the presence of liquids even when they're invisible.

From the results shown in figure 4 and in the video[1], it is clear that the LSTM CNN can at least roughly detect and track liquids, although its pixel-wise accuracy is not always 100%, especially when the liquid is not visible. Nevertheless, unlike the task of image segmentation, our ultimate goal is not to perfectly estimate the potential location of liquids, but to perceive and reason about the liquid such that it is possible to manipulate it using raw sensory data. For this, a rough sense of where the liquid is in a scene and how it is moving might suffice. Neural networks, then, have the potential to be a key component for enabling robots to handle liquids using robust, closed-loop controllers.

VII. Future Work

Further pursuing the problem of perceiving and reasoning about liquids, in future work we plan to combine the problems of detection and tracking into a single problem. Our goal is to not only perceive the liquid as it moves, but also to determine how much liquid is contained in the objects in the scene, and how much liquid is flowing. This is a necessary step before robots can apply control policies to liquid manipulation. The results here clearly show that the LSTM CNN is best suited for this task, and it is this type of network design we plan to investigate further for simultaneous detection and tracking of liquids from raw simulated imagery.

Another avenue for future work that we are currently pursuing is applying the techniques described here to data collected on a real robot. As stated in section III, it can be difficult to get the ground truth pixel labels for real data, which is why we chose to use a realistic liquid simulator in this paper. However, we are developing a method that uses a thermal infrared camera in combination with heated water to acquire ground truth labeling for data collected using a real robot. The advantage of this method is that heated water appears identical to room temperature water on a standard color camera, but is easily distinguishable on a thermal camera. This will allow us to label the "hot" pixels as liquid and all other pixels as not liquid.

Finally, we also plan to release not only our large dataset of labeled images, but also our code for generating this dataset. Other researchers will be able to apply their own algorithms to detecting and tracking liquid from raw sensory data. They will also be able to generate more data, and even vary how the data is generated (e.g., adding different types of cups, such as a glass cup). This will be the first dataset dedicated to perceiving and reasoning about liquids directly from raw sensory data generated via realistic simulation.

[1] Video of the full sequences at https://youtu.be/m5z0aFZgEX8

References

[1] Blender Online Community. *Blender - A 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2016. URL http://www.blender.org.

[2] Maya Cakmak and Andrea L Thomaz. Designing robot learners that ask good questions. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 17–24, 2012.

[3] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[4] Shane Griffith, Vladimir Sukhoy, Todd Wegter, and Alexander Stoytchev. Object categorization in the sink: Learning behavior–grounded object categories with water. In *Proceedings of the 2012 ICRA Workshop on Semantic Perception, Mapping and Exploration*. Citeseer, 2012.

[5] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 3338–3346, 2014.

[6] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain

tumor segmentation with deep neural networks. *arXiv preprint arXiv:1505.03540*, 2015.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[9] Carolin Körner, Thomas Pohl, Ulrich Rüde, Nils Thürey, and Thomas Zeiser. Parallel lattice boltzmann methods for cfd applications. In *Numerical Solution of Partial Differential Equations on Parallel Computers*, pages 439–466. Springer, 2006.

[10] Lars Kunze and Michael Beetz. Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. *Artificial Intelligence*, 2015.

[11] Joshua D Langsfeld, Krishnanand N Kaipa, Rodolphe J Gentili, James A Reggia, and Satyandra K Gupta. Incorporating failure-to-success transitions in imitation learning for a dynamic pouring task. In *IEEE International Conference on Intelligent Robots and Systems (IROS) Workshop on Compliant Manipulation*, 2014.

[12] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*, 2015.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[14] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *International Conference on Neural Information Processing Systems (NIPS)*, pages 2863–2871. 2015.

[15] Kei Okada, Mitsuharu Kojima, Yuichi Sagawa, Toshiyuki Ichino, Kenji Sato, and Masayuki Inaba. Vision based behavior verification system of humanoid robot for daily environment tasks. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, pages 7–12, 2006.

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

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

[18] Bernardino Romera-Paredes and Philip HS Torr. Recurrent instance segmentation. *arXiv preprint arXiv:1511.08250*, 2015.

[19] Leonel Rozo, Pedro Jimenez, and Carme Torras. Force-based robot learning of pouring skills using parametric hidden markov models. In *IEEE-RAS International Workshop on Robot Motion and Control (RoMoCo)*, pages 227–232, 2013.

[20] Minija Tamosiunaite, Bojan Nemec, Aleš Ude, and Florentin Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59 (11):910–922, 2011.

[21] Akihiko Yamaguchi and Christopher G Atkeson. Differential dynamic programming with temporally decomposed dynamics. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, pages 696–703, 2015.