Learning Robotic Manipulation of Granular Media

Connor Schenck*[A,B], Jonathan Tompson[B], Dieter Fox[A] and Sergey Levine[B,C]

[A]University of Washington
[B]Google, Inc.
[C]University of California Berkeley

Abstract: In this paper, we examine the problem of robotic manipulation of granular media. We evaluate multiple predictive models used to infer the dynamics of scooping and dumping actions. These models are evaluated on a task that involves manipulating the media in order to deform it into a desired shape. Our best performing model is based on a highly-tailored convolutional network architecture with domain-specific optimizations, which we show accurately models the physical interaction of the robotic scoop with the underlying media. We empirically demonstrate that explicitly predicting physical mechanics results in a policy that out-performs both a hand-crafted dynamics baseline, and a "value-network", which must otherwise implicitly predict the same mechanics in order to produce accurate value estimates.

Keywords: deep learning, manipulation, granular media

1 Introduction

Playing in the sand and constructing sand castles is a common pastime for children. It is easy for a child to view a pile of sand, visualize what they want to make, and construct a reasonable approximation to this desired shape. These kinds of skills transfer readily to a wide variety of tasks; from manipulating beans while cooking, to using construction machinery to excavate soil. In fact, it would seem that reasoning about manipulation of granular media and fluids is no more difficult for humans than manipulation of rigid objects. Yet in robotics there has been relatively little work examining how robots can learn such manipulation skills. If we want robots to operate intelligently in our environments, then it is necessary for them to be able to reason about substances like sand and beans that do not lend themselves readily to straightforward kinematic modeling.

In this paper, we examine methods for modeling granular media using either a convolutional network (ConvNet) architecture (in both dynamics or value-net variants), or using a fixed-function baseline method. The goal of this paper is to examine how these models can enable interaction with substances that cannot be modeled easily or accurately with simple kinematics, and whether or not an explicit dynamics prediction is necessary to solve our manipulation task. To that end, we evaluate the models on a task where the robot must use a scoop to manipulate pinto beans in a tray to form a desired shape (Figure 1 shows our experimental setup). This task domain combines multiple interesting challenges. Firstly, the beans are difficult to model as a set of discrete rigid objects and thus have non-trivial dynamics. Secondly, this domain allows for a wide variety of distinct manipulation tasks; such as pushing, collecting, scooping, or dumping to arrange the beans into a desired shape. Finally, this domain allows us to directly compare multiple models for manipulating granular media, from hard-coded heuristics to learned convolutional networks.

To solve this task, we develop an algorithm that uses learned predictive models to choose an action sequence which will form a target shape from the granular media. We study four such predictive models. *1.* A standard, unstructured ConvNet to predict the shape of the media conditioned on an initial configuration and a candidate manipulation action. *2.* A similar ConvNet model with additional task-specific optimizations built into the architecture to improve prediction accuracy. *3.* A

[*] Corresponding author: schenckc@cs.washington.edu

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

ConvNet to simply learn whether an action is “good” or “bad” with respect to a given goal shape. *4.* A baseline method which uses geometric heuristics to compute a rough estimate of the resulting arrangement of the media, allowing us to measure the value of learning the model rather than specifying it by hand.

We evaluate all of our predictive models on “easy” and “hard” target shapes. In both cases, the ConvNets trained to explicitly model the dynamics outperforms other methods. However, our results show that using a black-box ConvNet does not suffice; rather, policy performance is improved significantly when the network is structured in a way as to preserve essential physical properties of the dynamics. By using a network trained in a structured manner, the robot is able to accurately predict the state transitions and is better able to choose actions which maximize its potential to reach the goal shape.

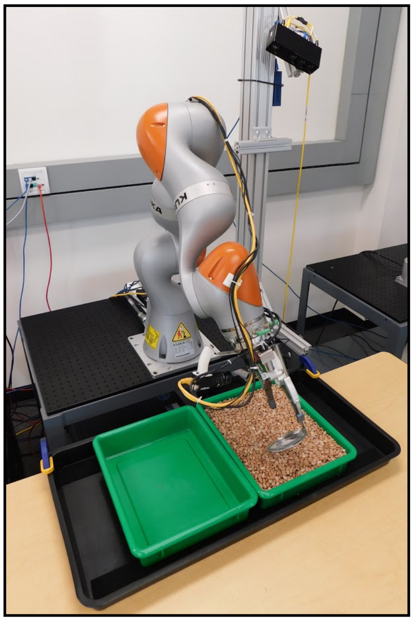


Figure 1: One of the robots in front of it’s table. In the gripper is the rigidly attached spoon used for scooping. On the table is the tray (green) containing the pinto beans used as the granular media. The black tray is to catch spilled beans.

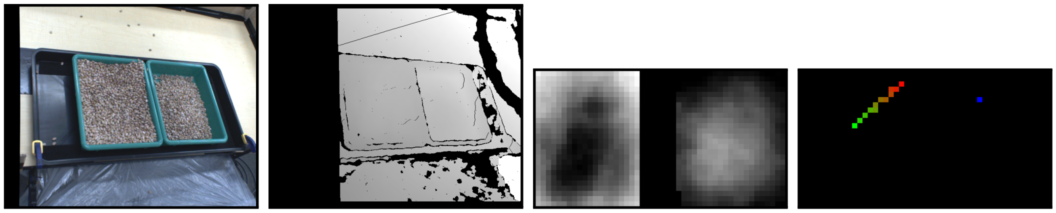
2 Related Work

In recent years, there has been some work in robotics in areas related to interaction and manipulation of granular media. For example, there has been a significant amount of work on legged locomotion over granular media [1, 2, 3]. There has also been work on automated operation of construction equipment for scooping [4, 5, 6, 7]. Additionally, much of the work related to robotic pouring has utilized granular media rather than liquids [8, 9, 10, 11, 12]. Recent work by Xu and Cakmak [13] explored the ability of robots to clean, among other things, granular media from surfaces using a low-cost tool. In contrast to this prior work, here we directly tackle the problem of manipulating granular media in a robust, flexible manner.

To manipulate granular media, our robot needs to develop a control policy. In this paper we focus on policies that utilize ConvNets either for direct policy learning or indirectly (e.g., by estimating dynamics). While there hasn’t been much prior work developing such policies for manipulating granular media, there has been work on developing control policies for tasks such as maze navigation [14], car racing [15], and playing common Atari video games [16]. However, all of these were performed in simulators, where it is easy to generate large amounts of data to train the underlying ConvNets. It is much more difficult to train deep learning algorithms like these in a real robot environment. Nonetheless, recent work has applied techniques similar to this using methods such as transfer learning from simulation to real-world [17, 18] or structured learning methods like guided policy search [19, 20]. These types of methods take advantage of other techniques (e.g., simulation or optimal control) to bootstrap their learning models and thus reduce the amount of data needed. However, an alternative approach is to simply collect more data. This has been carried out in the literature by utilizing an array of robots, rather than a single robot, and allowing them to collect data continuously over a long period of time [21, 22, 23]. We use this approach in this paper to allow us to collect a large dataset for training our learning models.

One of the main types of models that the robot learns in this paper is a predictive model using ConvNets. Recent work in robotics has shown how ConvNets can learn pixel-wise predictions for a video prediction task [23], as well as the dynamics of objects when subjected to robotic poke actions [24]. However, work by Byraven and Fox [25] showed that for dense, unstructured state spaces (in their case raw point clouds from a depth camera), it is often necessary to enforce structure in the network to achieve maximal performance. In this paper, we compare a standard unstructured network to a structured network for learning dense predictions involving granular media.

Similar to this work, ConvNets have recently been utilized for learning intuition of physical interactions by mimicking physics simulators [26, 27], for detecting and pouring liquids [28, 29], and for



(a) RGB (b) Depth (c) Height-Map (d) Action Map

Figure 2: An example data frame. From left to right: the RGB image, the corresponding depth image, the height-map computed from the depth image, and finally an example action map.

explicit modeling of rigid body dynamics [25, 30] or fluid dynamics [31]. To our knowledge, this work is the first to use ConvNets to predict the dynamics of granular media.

3 Task Overview

In this paper the robot is tasked with using a scoop to manipulate granular media in a tray to form a desired shape. Given an initial state of the environment h_0, and a goal state h_g, the robot must select a series of actions $a_0, ..., a_T$ that minimizes the L1-norm[2] between the final and goal states, i.e.,

$$l(h_0, a_0, ..., a_T, h_g) = \|\mathcal{F}(h_0, a_0, ..., a_T) - h_g\|_1$$

where $\mathcal{F}$ applies the actions $a_0, ..., a_T$ sequentially to the initial state h_0 to produce the final state $\mathcal{F}(h_0, a_0, ..., a_T) = h_{T+1}$. The state h_t is represented as a height-map over the media in the tray and is a 2D grid, where the value in each cell is the height of the surface of the granular media at that location in the tray. For this paper, each cell is approximately 1 cm × 1 cm. An example of a height-map is shown in Figure 2c.

3.1 The *scoop & dump* Action

The action the robot performs in this paper is a *scoop & dump* action, which takes a set of parameters θ_t and then performs a scoop on the bottom of the tray followed by a dump above it. The parameters θ_t comprise 6 elements (forming a 9D vector): the start location (2D), the start angle (1D), the end location (2D), the end angle (1D), the roll angle (1D), and the dump location (2D). The action is split into two parts, *scoop* and *dump*. At the start of the *scoop*, the robot moves the tip of the scoop to the XY coordinates specified by the start location in θ_t in the bottom of the tray. The robot then moves the tip of the scoop in a straight line along the bottom towards the XY coordinates specified by the end location. During the *scoop*, the robot sets the scoop angle, that is the angle between the plane of the tray and the plane of the scoop (with 0 being parallel), by linearly interpolating between the start and end angles specified in θ_t. Additionally, the robot sets the leading edge of the scoop (the part of the scoop pointing towards the end location) according to the roll angle, with the front leading when the roll angle is 0 degrees and the side of the scoop leading when the roll angle is 90 degrees. After the scoop reaches the end location, the robot pulls it directly up out of the tray. The robot then performs the *dump*, which consists of moving the scoop tip over the dump location and rotating the scoop to a fixed angle to allow any granular media on it to fall to the tray.

Note that the *scoop & dump* parameterization results in a diverse set of possible behaviors, and can represent complex manipulations in addition to just scooping and dumping; for instance a *"scoop"* with the spoon edge perpendicular to the line of motion will result in a *"pushing"* action, where no beens are collected in the spoon, but are rather deposited at the end of the scoop motion.

We reparameterize the set of action parameters, θ_t, as an "action map"; an image the same size as the height-map and which encodes the spatial location of our scoop and dump actions. Starting from an empty image with the same dimensions as the height-map, we draw a straight line from the start location to the end location, and we linearly vary the color of the line from red to green. Additionally we draw a blue dot at the dump location. Finally, we tile the 3 angles across 3 extra channels, which we concatenate channel-wise to the action map. An example of this action map (without the angles) is shown in Figure 2d. Representing the action parameters in this way brings

[2] We use the L1-norm rather than Earth mover's distance for the sake of simplicity.

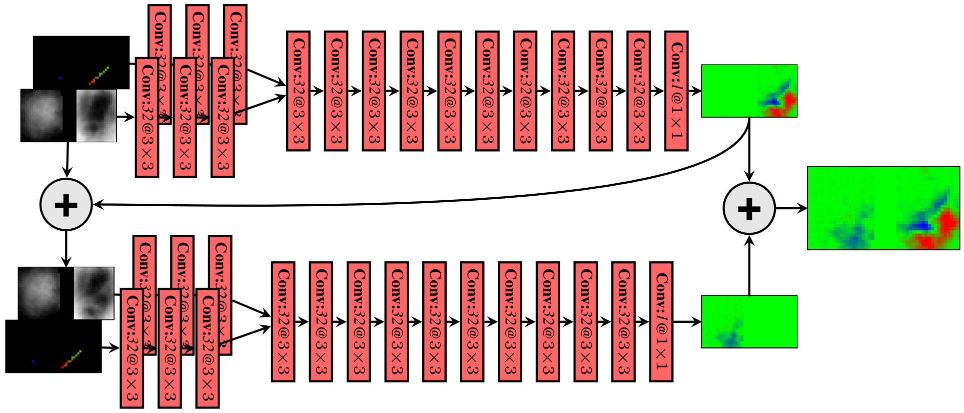


Figure 3: Network layout for the *scoop & dump*–net.

the actions into pixel-alignment with the ConvNets input state space (the height-map), which makes it more conducive for learning a stable and accurate mapping for our fully-convolutional network architecture (described in the following section).

4 Predictive Models

In this paper the robot's control policy that selects the series of actions $a_0, ..., a_T$ relies on an underlying predictive model, i.e., a model that is able to predict some future state based on the current state and a proposed action. We evaluate four different types of predictive models in this paper. The first two utilize ConvNets trained to predict the next state given the current state and a proposed action (both describe in the next section). The third utilizes a ConvNet to predict how much closer (or further) an action will take the robot from a given goal. The final model we add as a baseline comparison; it uses hard-coded heuristics to predict the next state from the current state and proposed action. The following sections describe each of these models. In section 6.1 we describe how the robot utilizes these models to select actions.

4.1 Predicting the Next State Using ConvNets

The first two predictive models take the current height-map h_t and a proposed action $\tilde{\theta}^i$ and attempt to predict the next state $\bar{h}_{t+1}$. To do this the robot uses one of two fully-convolutional network architectures: the *single*–net and the *scoop & dump*–net. The *single*–net is a standard fully-convolutional network (i.e. comprised of only convolution and ReLU layers) to predict the per-grid-cell change in the height-map (the layout of the *single*–net is identical to the top half of Figure 3). Empirically, this network tended to have difficulty conserving mass between the *scoop* and *dump* parts of the action (i.e., the mass *scooped* should be approximately equal to the mass *dumped*). To remedy this, we created the *scoop & dump*–net, which splits the network into two halves, one to predict the change from the *scoop* part of the action, and the other to predict the change from the *dump* part of the action. The *scoop & dump*–net is shown in Figure 3. The top half of the network takes in the current height-map and the action map, and outputs the predicted change due to the *scoop*. The bottom half of the network takes as input the current height-map *plus the changes predicted by the first network* and the action map, and predicts the changes due to the *dump*. In this way, the bottom half sees what the height-map is predicted to look like after the *scoop* has been performed. Additionally, to help the network enforce mass conservation, we add an explicit summation to the network that sums the change in mass due to the *scoop* and then passes it in a separate channel to the bottom half of the network, allowing it to directly reason about mass conservation. Finally, we also modify the loss function for training to include an additional term for the intermediate scoop state (recorded from real-world height-maps).

4.2 Predicting Distance to the Goal

Our third predictive model attempts to infer how much closer the next state will be to the goal compared to the current state. Specifically, given the current state h_t, the proposed action parameters

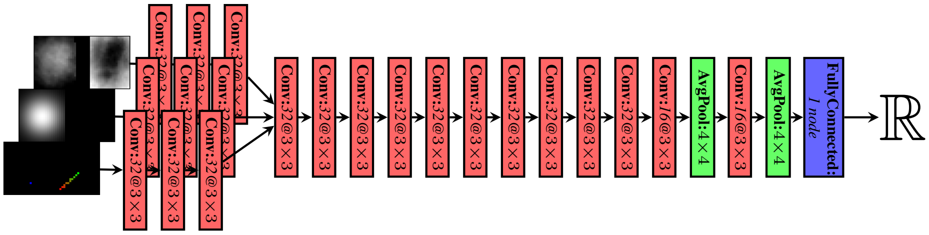


Figure 4: The network layout for the *value*–net.

$\tilde{\theta}^i$, and the goal state h_g, this model attempts to predict the following scalar quantity:

$$\|h_g - \mathcal{F}\left(h_t, a_t[\tilde{\theta}^i]\right)\|_1 - \|h_g - h_t\|_1$$

where $\mathcal{F}\left(h_t, a_t[\tilde{\theta}^i]\right)$ is the next state after applying action a_t with parameters $\tilde{\theta}^i$.

However unlike the previous two models of Section 4.1, this predictive model eschews explicitly computing the next state and instead uses a ConvNet to directly regress to the scalar difference in L1-norms. The network takes as input the current height-map h_t, the goal height-map h_g, and the proposed action map $\tilde{\theta}^i$, and outputs a single real value representing its prediction for the above equation. Intuitively this is a measure of "value" of a given action; a negative value will move the state closer to the target, while a positive value will move the state away from the target. The layout of the *value*–net is shown in Figure 4. When computing the scalar output (i.e. the "goodness" of a given action) the network is not required to explicitly predict the complex dynamics of the granular media. We compare this model to the other models in the results in section 7.

4.3 Baseline Predictive Model

Our final predictive model is a hard-coded heuristic function as a baseline for comparison. Similar to the first two predictive models from Section 4.1, this model predicts an approximate next state, conditioned on the current state h_t and a proposed action $\tilde{\theta}^i$. At a high-level, the model approximates a volume of mass to be removed from the height-map at the *scoop* location, and places the corresponding mass directly at the *dump* location. The details are as follows.

Given the start pose s^i and end pose e^i in $\tilde{\theta}^i$, the robot draws a straight line from s^i to e^i over the current height map h_t. Next the robot generates $h_{t+scoop}$ by removing all the granular media within $\frac{w}{2}$ cm of the line, where w is the width of the scoop, with the exception of media before s^i or after e^i. This results in a rectangular section of $h_{t+scoop}$, running along the line segment from s^i to e^i, set to all 0s (i.e., all the granular media removed).

The robot next computes the sum total of the volume of media removed, and divides it between c, the volume of media *scooped*, and p, the volume of media *pushed*. For every grid cell in the rectangle affected by the *scoop* action described in the previous paragraph, the robot computes α_g, the angle of the scoop. This angle α_g is computed by linearly interpolating between the start angle, α_s^i, and the end angle, α_e^i, along the line from s^i to e^i (both α_s^i and α_e^i are specified by the proposed action $\tilde{\theta}^i$). Recall from Section 3.1 that the angle of the scoop is the angle between the plane of the tray and the plane of the scoop, with 0 degrees being parallel. For grid cells where α_g is positive, their volume is added to c, and for grid cells where α_g is negative, their volume is added to p. Intuitively, this means that when the open face of the scoop is facing forward (i.e., α_g is positive), the media is *scooped*, that is, it stays in the scoop when it is raised from the tray. However, when the underside of the scoop is facing forward (i.e., α_g is negative), the media is *pushed*, that is, it is pushed in front of the scoop until it reaches e^i, then stays there and does not go with the scoop when it is raised.

Finally, to generate h_{t+1}, the robot adds two narrow Gaussians to $h_{t+scoop}$. It adds the first at e^i and adjusts it so that the volume of the Gaussian is equal to p. This is the media that was *pushed* by the scoop. It adds the second at the dump location specified in $\tilde{\theta}^i$ and adjusts it so that its volume is equal to c. This is the media that was *scooped* by the scoop, and thus is deposited by the scoop at the dump location.

5 Experimental Setup

5.1 Robot and Sensors

In this paper we utilized 7 KUKA LBR IIWA robotic arms, however each acted independently and did not interfere with each other. The use of multiple arms in parallel allows us to perform data collection at a faster rate. The arms have 7 degrees of freedom and are each mounted on a horizontal table. Another table is placed immediately in front of each robot arm and acts as the robot's workspace. The robotic setup is shown in Figure 1. Each robot has it's own depth camera capable of recording 640×480 depth images at 30 Hz.

5.2 Objects and Granular Media

We use pinto beans as the granular media since they are both large enough to not lodge in the robot's joints and small enough to have interesting dynamics. They also appear with minimal noise on infrared depth cameras and are rigid enough to not break or deform after extended manipulation. For the scoop tool we use a standard large metal cooking spoon, which we rigidly attach to the robot's end-effector. Finally, we place a tray in front of the robot and rigidly fix it to the table. The tray we use has a divider that splits the tray into left and right halves, which allows for more interesting interactions with the granular media by the robot. The precise pose of the tray relative to the robot and the camera are calibrated *a priori* and fixed for the duration of the experiments.

5.3 Data Collection

To train the ConvNets used by the robot policy, we collected a dataset of robot scooping actions. First, we filled one side of the tray in front of each robot with 3.75kg of pinto beans. Next, each robot observed the tray in front of it and generated a goal state by randomly placing a target pile in one of the four corners or center of the empty side of the tray. The robots then executed their policies, attempting to scoop the beans from the full side of the tray to the target pile on the empty side. After 75 iterations of the *scoop & dump* action, the robots randomly generated a new goal on the other side of the tray. This process of generating goals and executing the policy was repeated until sufficient data had been collected, with the experimenter periodically resetting all the beans to one side of the tray in between repetitions.

Overall, we collected approximately 15,000 examples of the *scoop & dump* action. For the first 10,000, the robots used the baseline scoring function during policy execution as described in section 4.3. For the last 5,000, we trained a *scoop & dump*–net on the first 10,000 data-points and then used the dynamics scoring function for the policy as described in section 4.1.

5.4 Data Processing

The state representation used by the policy in this paper is a height-map over the media in the tray, however the robots' observations are depth images from a depth camera. To convert the depth images to a height-map, the robot first constructs a point-cloud from the depth image using the camera's known intrinsics. The point-cloud is then transformed into the tray's coordinate frame and points within the tray's bounding-box are projected to the 2D discretized height-map surface, recording the resultant height above the tray. Figures 2b and 2c show a depth image and its corresponding height-map. The robot's arm and the scoop are excluded from the image by moving the arm up and out of the robot's view.

6 Evaluation

6.1 Robot Policy

To select the correct actions, the robot utilizes a greedy policy based on the cross-entropy method (CEM) [32]. CEM is a sampling method that uses a scoring function L to iteratively sort samples, estimate a distribution from the top K, and resample. To compute the action parameters θ_t, the robot first samples N sets of parameters $\tilde{\theta}^1, ..., \tilde{\theta}^N$ uniformly at random from the parameter space. Next, for each sampled parameter set $\tilde{\theta}^i$, the robot computes the score $L(\tilde{\theta}^i, h_t, h_g)$ using the scoring function L, the current state h_t, and the goal state h_g. Intuitively, L is a measure of how “good” the robot believes taking an action with parameters $\tilde{\theta}^i$ in state h_t towards goal h_g will be, with lower values indicating better parameters. The robot then discards all but the best K sets (i.e., the lowest scoring K sets), where $K < N$. Next the robot fits a Gaussian distribution to the K samples, and then resamples N new sets of parameters from this Gaussian. The robot repeats this loop of sampling from a distribution, picking the best K, and refitting the distribution, for a fixed number of steps. On the final iteration, the robot sets θ_t to be the values of the top performing set of parameters $\tilde{\theta}^i$.

The scoring function L uses the predictive model underneath to compute the score. For the models in sections 4.1 and 4.3, since they both predict the next state, the scoring function simply returns the L1 error (mean absolute error) between the predicted next state and the goal state. For the model described in section 4.2, the robot directly uses the output of the network as the score since it already predicts how “good” a proposed action is. For the results in section 7, we compare using each of the predictive models in the scoring function.

6.2 Training the Models

The predictive models described in sections 4.1 and 4.2 utilize ConvNets. To train those networks, we use the dataset collected in section 5.3. For all networks, we use the mini-batched gradient descent method Adam [33] with a learning rate of $5\mathrm{e}{-4}$ to iteratively update the weights of the networks. All weights were initialized randomly. We first pre-trained all the networks for 30,000 iterations using the baseline model output predictions (using initial states from the robot data) described in section 4.3[3]. Next we trained the networks for 100,000 iterations on the initial and next states captured from the robot.

Since each network was slightly different from the others, we used different loss functions for each. For the *single*–net (described in section 4.1), we use the L2-loss on the predicted next state. For the *scoop & dump*–net (also described in section 4.1), we also use the L2-loss on the predicted next state, but we add another L2-loss on the output of the top half of the network (the *scoop* only prediction), which is added with equal weight. Additionally we stop the gradients from the first loss from propagating directly back into the top half of the network, which encourages the top half to predict only the result of the *scoop* part of the action. Finally, for the *value*–net (described in section 4.2) we use an L2-loss on the change in mean absolute error to that goal from the before action state to the after action state (each scoop action in the dataset had a corresponding goal).

6.3 Tasks

We evaluate the robot’s models on two example tasks, a “simple” task and a “hard” task. The first task is for the robot to scoop on one side of the tray that is filled with beans, and dump the beans into a Gaussian-shaped pile on the empty side. The robot is given 100 *scoop & dump* actions to complete the goal shape. While this may seem relatively simple, it is actually quite complicated. The robot must be able to reason about what types of scoop actions will result in beans collecting in the scoop. It also must be able to reason about how many beans it will acquire and about where those beans will go when dumping them. That is, for even such a “simple” task the robot must have an intimate understanding of the dynamics of the granular media. We also evaluate the robot on a “hard” task that adds another level of complexity. In this task, the robot must dump the beans into a more complex shape, requiring the robot to reason in more detail about the dynamics of the dumped beans. In this paper, we use the shape of the letter “G” (in Baskerville typeface) as the target.

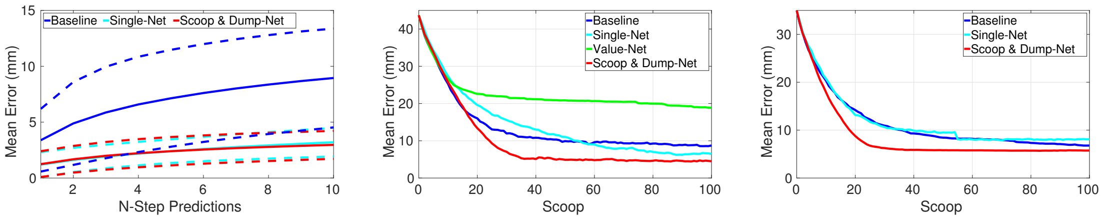
7 Results

Figure 5 shows the results[4]. Figure 5a shows the error between predicted next states and the actual next states on a test set for each of the three models that predict state transitions [5]. The X-axis shows

[3] We empirically determined that the networks performed better with pre-training than without.

[4] Please refer to the associated video for additional results: `https://youtu.be/YUb5hi-OK70`

[5] For this portion of the analysis we don’t compare with the *value*–net because there is no next-state prediction to compare with, although empirically the error for it did seem to converge to a reasonable value after training.



(a) Test Set Error (b) Pile Task Error (c) G Task Error

Figure 5: The results of evaluating our predictive models. Left to right: mean error (in mm, confidence regions indicated by dashed lines) of the 3 models predicting the state transition; the mean error of the 4 models on the pile task and on the G task.

how the models perform over multi-step predictions (i.e., using their own output as input for the next timestep), and the Y-axis shows the average error in millimeters, with the dashed lines indicating the standard deviation for each model. Interestingly, the *single*–net and the *scoop & dump*–net perform very similarly on our test set. As expected, the rough heuristic baseline model performs poorly as it does not attempt to accurately predict the dynamics but only roughly captures the overall behavior.

From Figure 5b it is apparent that while the baseline does not predict accurate dynamics, it is never-the-less able to solve the pile task to some degree. Figure 5b shows the error in millimeters for each model averaged across 3 runs for the pile task with each run initialized to the same mass of beans (3.75kg) and leveled in the same container. The X-axis shows the error over time as the robot completes more *scoop & dump* actions (100 total). In all cases the error goes down over time as expected. From the graph it is apparent that the *value*–net performs much more poorly than the other models and is unable to reason effectively about the robot's actions. Without explicitly training the network to learn the rich, high-dimensional statistics of the full output state representation (as in the *scoop & dump*–net), it may easily get stuck in local minima during training (particularly in the regime of limited training data) and as a result performs poorly. It is clear that it is unsuitable for these kinds of tasks and so we leave it out of the remaining evaluations. Interestingly, the baseline method outperforms the *single*–net method for the first 50 of 100 actions, and although it does eventually converge to a state closer to the goal, it does so much more slowly. The *scoop & dump*–net, on the other hand, not only reaches a state closer to the goal, but converges much faster than any other method. This clearly indicates that even though the *single*–net and *scoop & dump*–net had similar performance on the test set, when actually using the models on control tasks, the *scoop & dump*–net is better able to inform the policy about the dynamics of the actions on the granular media.

Examining Figure 5c, it is clear that this trend remains true for even more complicated tasks. This figure shows the performance of 3 of the models on the G task (we left out the *value*–net due to its poor performance on the pile task) averaged across three runs each. Interestingly, the baseline, while converging more slowly in this case than the *single*–net, in the end outperforms it by a narrow margin. These results taken together indicate that the structure of the *scoop & dump*–net is better suited to acting as a predictive model of a robotic control policy involving granular media than a naive unstructured network like the *single*–net.

8 Conclusion

In this paper, we developed 4 predictive models for enabling a robotic control policy to manipulate granular media into a desired shape. We evaluated these models on both a test set and on a pair of manipulation tasks of varying difficulty. The results clearly show that for tasks involving granular media, the best performance can be achieved by training structured ConvNets to predict full state transitions. Our *scoop & dump*–net, which explicitly modeled the different parts of the action in the network structure, outperformed both our *value*–net, which only predicted the value of an action with respect to a goal, and our *single*–net, which predicted the full state transition but did not have any internal structuring to reflect the nature of the action. This is consistent with prior work [25], which showed that structured networks are also necessary when predicting the motion of dense point clouds. Our results show using structured, learned predictive models can enable robots to reason about substances such as granular media that are difficult to reason about with traditional kinematic models.

Acknowledgments

The authors would like to thank Peter Pastor for his invaluable insights and patient help.

References

[1] C. Li, P. B. Umbanhowar, H. Komsuoglu, D. E. Koditschek, and D. I. Goldman. Sensitive dependence of the motion of a legged robot on granular media. *Proceedings of the National Academy of Sciences*, 106(9):3029–3034, 2009.

[2] C. Li, P. B. Umbanhowar, H. Komsuoglu, and D. I. Goldman. The effect of limb kinematics on the speed of a legged robot on granular media. *Experimental mechanics*, 50(9):1383–1393, 2010.

[3] C. Li, T. Zhang, and D. I. Goldman. A terradynamics of legged locomotion on granular media. *Science*, 339(6126):1408–1412, 2013.

[4] A. Hemami. Motion trajectory study in the scooping operation of an lhd-loader. *IEEE Transactions on Industry Applications*, 30(5):1333–1338, 1994.

[5] T. Takei, K. Ichikawa, K. Okawa, S. Sarata, T. Tsubouchi, and A. Torige. Path planning of wheel loader type robot for scooping and loading operation by genetic algorithm. In *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*, pages 1494–1499. IEEE, 2013.

[6] O. Kanai, H. Osumi, S. Sarata, and M. Kurisu. Autonomous scooping of a rock pile by a wheel loader using disturbance observer. *ISARC*, 2006.

[7] S. Sarata, H. Osumi, Y. Kawai, and F. Tomita. Trajectory arrangement based on resistance force and shape of pile at scooping motion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3488–3493. IEEE, 2004.

[8] L. Rozo, P. Jimenez, and C. 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.

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

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

[11] A. Yamaguchi and C. Atkeson. Differential dynamic programming for graph-structured dynamical systems: Generalization of pouring behavior with different skills. In *Proceedings of the International Conference on Humanoid Robotics (Humanoids)*, 2016.

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

[13] Z. Xu and M. Cakmak. Enhanced robotic cleaning with a low-cost tool attachment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2595–2601, 2014.

[14] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[15] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2017.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[17] F. Sadeghi and S. Levine. (cad)2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[18] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2015.

[19] S. Levine and V. Koltun. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1–9, 2013.

[20] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[21] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, page 0278364917710318, 2016.

[22] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2017.

[23] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2017.

[24] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[25] A. Byravan and D. Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2017.

[26] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016.

[27] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[28] C. Schenck and D. Fox. Towards learning to perceive and reason about liquids. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2016.

[29] C. Schenck and D. Fox. Visual closed-loop control for pouring liquids. In *Proceedings of the International Conference on Experimental Robotics (ICRA)*, 2017.

[30] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016.

[31] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating eulerian fluid simulation with convolutional networks. *ICML*, 2017.

[32] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[33] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.