This is the accepted version of a paper presented at 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE).

Citation for the original published paper:

Mitsioni, I., Karayiannidis, Y., Kragic, D. (2021)
Modelling and Learning Dynamics for Robotic Food-Cutting
In: (pp. 1194-1200).
https://doi.org/10.1109/CASE49439.2021.9551558

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-305257
Modelling and Learning Dynamics for Robotic Food-Cutting

Ioanna Mitsioni¹, Yiannis Karayiannidis² and Danica Kragic¹

Abstract—Interaction dynamics are difficult to model analytically, making data-driven controllers preferable for contact-rich manipulation tasks. In this work, we approximate the intricate dynamics of food-cutting with a Long Short-Term Memory (LSTM) model to apply a Model Predictive Controller (MPC). We propose a problem formulation that allows velocity-controlled robots to learn the interaction dynamics and tackle the difficulty of multi-step predictions by training the model with a horizon curriculum. We experimentally demonstrate that our approach leads to good predictive performance that scales for longer prediction horizons, generalizes to unseen object classes and results in controller behaviors with an understanding of the cutting dynamics.

I. INTRODUCTION

Modelling and learning dynamics for contact-rich manipulation is an open problem in robotics. Classical control approaches [1]–[5] suffer when the modes of interaction increase, their respective models are too complicated to be described analytically or their variations too diverse to be accounted for. Especially for contact-rich tasks, this difficulty arises from the dynamics that can include discontinuities such as breaking and making contact, complicated frictional phenomena, or the variety of object properties. With the introduction of data-driven methods, a lot of these shortcomings were confronted successfully [6]. Their main advantage stems from not relying on analytical models, but on interaction with the environment, or demonstrations that can initialize a policy for the completion of the task.

In this work, we investigate a data-driven method for robotic food cutting which is inherently a contact-rich task with complicated interaction dynamics. Cutting dynamics can vary spatially and temporally for the same object, as most food items are not structurally homogeneous; for the same object class, as not two objects within a class have the same size or shape; and between classes, as food items are very diverse. Modelling the interaction as a mass-spring-damper system is an oversimplification of the contact dynamics and the tissue fracturing/separation of the fibers is not well-approximated by a smooth impedance in closed-form expression. On the other hand, more realistic and analytical representations [7]–[9] are arduous to develop when considering many different classes, or classes with substantial variations in their dynamics. As a task, cutting can be low-dimensional when expressed in the operational space. The discontinuities it exhibits are in most cases due to frictional, stick-slip phenomena and not extreme ones such as sudden and complete breaking of contact. However, it is exceptionally difficult to simulate, so any type of data collection or exploration must be done on the real system, which is expensive. As a result, we choose to learn only the dynamics model from data with a deep network and handle the closed-loop control with a Model Predictive Controller (MPC) as seen in Fig. 1.

For methods that are not learning a policy online, as model-free Reinforcement Learning techniques or end-to-end MPC variants [10], the resulting controller primarily depends on the accuracy and expressive capacity of the dynamics model. In real systems, the available quantities are constrained by hardware and perception capabilities, and data-driven methods need to adapt to that. For example, the collected dataset for a passively compliant, torque-controlled robot can differ significantly from that of a rigid, velocity-controlled one. Additionally, while model-based approaches such as MPC can be sample-efficient, they hinge on accurate multi-step predictions, which is not a trivial problem [11].

In this paper, we present a velocity-resolved formulation for contact-rich tasks and demonstrate it in learning food-cutting dynamics. We build upon our previous work [12] and reduce the assumptions on the formulation of the learning problem. Furthermore, we propose a new model training scheme that simplifies training and increases the accuracy of multi-step predictions. In our experiments, we show that the proposed modelling and training approach can provide models that have consistently good performance and exhibit a fine-grained understanding of the task dynamics when performing the task within an MPC.

II. RELATED WORK

Robotic cutting has been treated in a multitude of ways, primarily based on more traditional control approaches. In [13] the authors employed an impedance controller with
adaptive force tracking for a simulated object with non-homogeneous stiffness. An adaptive position controller with gradient-based estimation of the desired force was presented in [14] with simulated results as well. More recently, [9] proposed force control combined with visual servoing to adjust the tracked trajectories for the task of cutting deformable soft objects, while minimizing the required cutting force. A combined hybrid force/position control approach was presented in [8] to cut two classes of non-deformable objects. These works all introduce physics-based mechanics that lead to a well-defined problem. However, their applicability is limited as they require further computations to be applied to a larger variety of cases, as opposed to Deep Networks that can generalize.

Data-driven approaches can address this issue by approximating the interaction dynamics, resulting in a single model that is capable of treating several object classes. A method that employs deep networks to approximate the dynamics for this task was first introduced in [15]. Although the method’s performance was evaluated on an extensive dataset, the generalization ability to unseen classes was not examined. Additionally, the proposed network outperformed several baselines but it was not clear whether there was need for the complex architecture and training procedure, as only one architecture was evaluated within the MPC. This approach was revisited by our group in [12] after being reformulated into a velocity resolved control problem, but with the same underlying network structure and training procedure that consisted of hand-picking layers to initialize for training. Unseen classes were included in the evaluation but there was no further examination of the network and its training.

Other data-driven approaches to cutting include [16] where the focus is how to learn a semantic representation of the task rather than the dynamics. In [17], the authors used a saw instead of a knife to cut through solid objects. Using VAEs, they produced latent spaces for reconstruction and prediction, but the goal was cutting detection as well as material and thickness classification. Lastly, modelling of fracture and deformation in cutting has been examined in [18] through Finite Element Method (FEM) to predict the force and torque during slicing.

Despite the limited amount of works for this particular application, data-driven methods in contact-rich scenarios have shown promising results. Demonstration-based methods [19]–[22], are well suited due to their sample complexity, but infeasible for cutting with force feedback as there is no practical way to distinguish the demonstrator’s exerted wrench from the object’s. Reinforcement Learning, when actively focusing on sampling complexity, is a competitive alternative for real-world, contact-rich tasks. Recently, in [23] the authors proposed an actor-critic that is guided by supervised learning to account for sample complexity and safety, but still required 1.5 hours for an assembly task that has a smaller range of dynamics than cutting. Another method that reduces the sample complexity was presented in [24]. The authors actively leveraged a hand-engineered controller as a basis for a policy they optimize online, thus splitting the problem into a trajectory tracker and an adaptive corrective behavior. Their method greatly reduced the sample requirements through sim-to-real transfer but still depended on a simulated environment and unsupervised exploration, neither of which are available for our task.

A central part of the suggested training approach is curriculum training [25] which we combine with learning rate decay to avoid prediction error accumulation and facilitate training. Curriculum training has been applied in several different contexts but to the best of our knowledge, not as a horizon curriculum for dynamics model prediction. In this paper, we take inspiration from the work in [26] where it is used for image registration and the authors gradually increase the temporal distance between the images, and apply the same logic to increasingly distant prediction horizons. Other applications of curriculum training include mini-batch frequency selection [27], sequence prediction in natural language processing [28], equation learning [29] and finally, encoding positions and velocities from pixels in simulated control tasks [30].

III. PROBLEM DESCRIPTION

Cutting can be defined as a series of (often periodic) motions that apply an acutely directed force, until an object is separated. In this work, we are interested in sawing and slicing motions that enable the knife to reach the cutting board. In order to not damage the environment and itself, we want the robot to execute the motions while exerting appropriate force for the object at hand. The amount of exerted force is affected by how rigidly the robot is moving, how aggressive its motions are and naturally, by the dynamics of the object. For example, to successfully cut through a bell pepper and not crush it, the robot needs to employ long sawing motions to break through the skin and then simply slice downwards. In contrast, this technique is insufficient to cut a potato which requires more rigorous sawing to counteract the friction of its stiff interior. Determining the correct parameters that encode the different behaviors requires either perfect knowledge of the dynamical model, or very tedious tuning procedures. Instead, we aim to learn the model and let the MPC optimize the control inputs based on it as described in the following sections.

A. Modelling Cutting

Consider a velocity-controlled manipulator equipped with force sensing. Let $p \in \mathbb{R}^3$ denote the translation part of the robot’s end-effector pose in the world frame and $f_e \in \mathbb{R}^3$ for the force measurements. To describe contacts, the majority of viscoelastic contact models $\mathcal{F}$ are based on a mapping between forces $f$ and displacements of the environment $\delta$ as an outcome of the force exerted by the robot tool: $f = \mathcal{F}(\delta, \dot{\delta})$. Interacting with static environment and assuming bilateral constraints (contact maintenance), allows to assume that $\delta = p$, yielding a forward dynamics model $\mathcal{G}$ for the position of the robot’s end-effector, $p_{t+1} = \mathcal{G}(p_t, f_t)$, where $f_t$ is the force exerted by the robot to the environment to deform it. However, a cutting task includes unilateral constraints,
break of contact and penetration of the constraints implying both plastic and elastic displacements \( z \). Thus, the contact force model

\[
\mathbf{f} = \mathcal{F}(z, \dot{z}, \dot{p})
\]

where \( z \) is the elastic deformation derivative for which \( \dot{z} = \dot{p} \) is no longer valid.

To account for the dependence on the hidden state \( z \), the forward dynamics model for the position of the robot’s end-effector needs to be augmented by including the contact force \( \mathbf{f}_{st} \) as an additional input:

\[
\mathbf{p}_{t+1} = \mathcal{G}(\mathbf{p}_t, \mathbf{f}_{st}, \mathbf{f}_t).
\]  

(1)

Note that the model can be expanded to predict the future forces, however as we will discuss in the control formulation, we are mainly interested in prediction of future positions.

**B. Controller Formulation**

We can manage the robot’s compliant behavior and command trajectories using the (inverse) damping controller for the robot’s translation velocities:

\[
\mathbf{u} = \mathbf{K}_a(\mathbf{f}_s - \mathbf{f}_r).
\]  

(2)

Then, to follow a trajectory of desired positions and velocities, \( \mathbf{p}_d, \dot{\mathbf{p}}_d \), the reference force exerted by the robot to the environment is:

\[
\mathbf{f}_r = \mathbf{K}_a^{-1}(\mathbf{K}_p \mathbf{e} - \dot{\mathbf{p}}_d)
\]  

(3)

where \( \mathbf{K}_p, \mathbf{K}_a \in \mathbb{R}^{3 \times 3} \) are the stiffness and compliance gain matrices and \( \mathbf{e} = \mathbf{p} - \mathbf{p}_d \) the position error.

However, this controller is not optimal. Planning the desired cutting trajectory \( \mathbf{p}_d, \dot{\mathbf{p}}_d \) and choosing the appropriate controller gains \( \mathbf{K}_a, \mathbf{K}_p \) depends not only on the object class, but the specific geometry and dynamics of each individual object. Instead, we encode the desired behavior through a cost function and employ an MPC to minimize it. The main components of the cost function are a term that drives the slicing motion towards the table’s surface \( \mathbf{p}_{table} \) and a sawing term that enables the downward progress. Since there is no fixed trajectory, the sawing term does not penalize motion within a range \( d \) around the central sawing point \( \mathbf{p}_{center} \) and is quadratic beyond it. Finally, to motivate smaller-effort solutions, we include the norm of the control input. Namely, for the prediction horizon \( H_t \), the cost is given by:

\[
C(\mathbf{p}, \mathbf{f}_r) = c_{cut} \sum_{k=1}^{H_t} (\mathbf{p}_k^0 - \mathbf{p}_{table})^2
\]  

\[
+ c_{saw} \sum_{k=1}^{H_t} \left( \max \{0, |\mathbf{p}^g_k - \mathbf{p}_{center}| - d \} \right)^2
\]  

\[
+ c_v \sum_{k=1}^{H_t} \|\mathbf{f}_{r,k}\|^2
\]

where \( c_{cut}, c_{saw} \) are positive constants weighting the contribution of the costs associated with cutting and sawing actions respectively to the total cost while \( c_v \) is the weighting constant for the control input cost term.

Since we do not have an analytical contact model, we approximate the forward dynamics with a deep network and determine the optimal reference force over a time horizon \( H_t \) by solving the optimization problem:

\[
f_r^* = \arg \min_{f_r} \sum_{k=0}^{H_t} C(\mathbf{p}_{t+k}, \mathbf{f}_{r,t+k}).
\]  

(4)

**IV. METHOD**

**A. Data-driven Modelling**

Our goal is to learn a dynamics model for an MPC that does not require visual information about the object, its size, shape or type. Concretely, we are interested in representing the interaction dynamics between the manipulator and the object as the transition function in Eq. (1). We parametrize the dynamics function as a deep network that receives current positions, measured and reference forces, and outputs the estimated future positions. We define the model’s state as the augmented state vector \( \mathbf{x}_t = [\mathbf{p}_t^T, \mathbf{f}_{r,t}^T] \) and denote \( \mathbf{v}_t = \mathbf{f}_{r,t} \).

Instead of velocities that are usually noisy and difficult to learn from, we employ relative displacements over time, similar to [12], [15]. To achieve that, the input features for the learning module are not treated as single time-steps but form non-overlapping sequences, called blocks. Denoting the system state expressed in blocks as \( \mathbf{X} \), block \( b \) of length \( M \) is then given by:

\[
\mathbf{X}^M_b = [\mathbf{x}_{bM}^T; \ldots; \mathbf{x}_{(b+1)M-1}^T], \quad \in \mathbb{R}^{M \times 6}.
\]  

(5)

If we denote the positional elements of \( \mathbf{X}^M_b \) as \( \mathbf{p}^M_b \) and an all-ones vector of length \( M \) as \( \mathbf{1}^M \), the transformation from positions to relative displacements is done by subtracting the past block’s last position from every position in the current one:

\[
\Delta \mathbf{p}^M_b = \mathbf{p}^M_b - \mathbf{1}^M \mathbf{p}_{(b-1)M-1}.
\]  

(6)

Dropping the superscript \( M \) for brevity, the network’s input is then \( \mathbf{X}_b = [\Delta \mathbf{p}_b, \mathbf{f}^*_b] \). Through this transformation, we also ensure that the network will not overfit to absolute positions, which do not carry the same amount of information as they depend on the object’s size. Since we are using relative displacements and sample every 5ms, the magnitude of the positional part is significantly smaller than the remaining features of the input vector. To ensure consistency in the input range, we normalize the features to zero mean and unit standard deviation.

In contrast to our earlier work [12], we model the effect of the control input on the system explicitly through the reference force and do not assume that it is measurable through the result of the interaction force. Expressing the reference force as a function of the desired velocity and the position error results in a more clear and concise formulation that offers a better representation for the learning task. To showcase this, we can transform the initial data space that includes multi-step sequences of 6 or 9 features, into a 2-D one and visualize the data with t-SNE [31]. t-SNE is a
proportional probabilistic dimensionality reduction technique that projects data into their low-dimensional embedding in a non-linear way, while trying to preserve their probabilistic distribution. To have a fair comparison, we used the same dataset but omitted the $f_i$ inputs for the latter case. The two resulting datasets went through the same pre-processing as in Eq. (5) and (6).

Fig. 2a shows the results of the dataset $D_1 = \{p, f_i\}$ and correspondingly, Fig. 2b the ones from the dataset we propose for this task, namely $D_2 = \{p, f_i, f_r\}$. With dataset $D_1$, as seen in Fig. 2a, there is no specific structure in the embedding except for the eggplant class, as the class dynamics are the most easily distinguishable during the task due to the object’s texture. In comparison, adding $f_r$ and visualizing $D_2$, produces more coherent clusters. The central part of the plot is mostly occupied by easier to cut classes and as we are moving peripherally outwards, we get cases of stiffer materials. Although we are not interested in classifying the objects, a more cohesive embedding indicates that $D_2$ is a more informative representation and henceforth, the networks we compare in the experimental section are trained on these features.

B. Network Architecture and Training

In this work, we chose to employ an LSTM network as opposed to Recurrent Neural Networks (RNN) used in [12], [15]. While more complex than a regular RNN, LSTMs have proven to be suitable for learning sequences and dependencies further in time [32], which is appealing for a task that requires modelling of temporally and spatially varying dynamics. The LSTM networks consist of a fully connected input layer of size 90 with a hyperbolic tangent activation, followed by 2 LSTM layers of hidden size 9 and a linear output layer that transforms the LSTM output to size 30. The RNN baseline consists of 6 fully connected layers with hyperbolic tangent activation and 2 recurrent layers with 30 units each.

To obtain the multi-step predictions required by the controller, we can either train the network to directly predict all the steps until the horizon $H_t$, or use Eq. (1) to iteratively predict one step ahead until we reach the desired horizon. Both problems result in sequence-to-sequence predictions, but the former is significantly more difficult to learn and since the loss will be averaged over the whole horizon, it is not guaranteed that the model will be accurate enough for the individual steps. The iterative prediction is a simpler problem to solve and computing the loss of all the intermediate steps helps the model capture the dynamics more accurately.

Considering the block formulation, for a robot working at 200Hz and with block length $M = 10$, instead of predicting to $H_t = 0.15s$, the new horizon will be $H_b = 0.15/0.05 = 3$ blocks ahead. Then, using the iterative method, we predict the future displacements using the intermediate prediction $\hat{p}_{b+1}, i \in [1, H/M]$ as inputs until we reach the desired horizon i.e.

$$\hat{p}_{b+1} = G(X_b, v_b)$$
$$\hat{p}_{b+2} = G(X_{b+1}, v_{b+1})$$
$$\ldots$$
$$\hat{p}_{b+H_b} = G(X_{b+H_b−1}, v_{b+H_b−1}).$$

A common problem with the iterative approach is error accumulation, since predictions are used in place of observations. To tackle that, we propose to train the system with a curriculum strategy that gradually increases the difficulty of the prediction goal. Practically, this amounts to progressively predicting further ahead in the future by increasing the horizon. However, the abrupt difference in difficulty might lead the system into instability, or it might render the hyperparameters used for the easier problem unsuitable. Therefore, we apply learning rate decay when the horizon changes, so that learning is adjusted to the new horizon smoothly and the gradient steps are affected less by the change, especially during the transitions. For every horizon, we train the network for 10 epochs and reduce the learning rate by a factor $gamma$ (see Table I), except for the final length prediction that we allow the network to train for 20 epochs without further changing it, in order to fine-tune its predictions for the final horizon.

C. Model Predictive Control

We treat the problem in Eq. (4) with an MPC that instead of solving the optimization problem for an infinite horizon, executes the first step of the solution and then re-samples the current state. We manage the compliant reaction to the environment separately as seen in Fig. 1, and use $f_r$ as a feature for the dynamics model and the optimization variable. To solve Eq. (4) we use a shooting method [33]. For every optimization iteration, we generate 25 potential inputs that act as the feasible forces for this round, query the dynamics model for the future states through the iterative approach in Sec IV-B and finally choose the one associated with the lowest cost as $f^*$. Lastly, for the MPC state we do not consider the full pose of the end-effector, but simplify the problem by only treating the translational parts of the cutting motion (axes $Y, Z$ in Fig. 1). The motion on the remaining axes is controlled through a set-point stiffness controller.

V. Evaluation

For all of the following experiments, the training set, or seen classes, includes trials for 6 object classes, $D_{seen} =$
function of their size, as to include enough samples of free-space motion.

In Sections V-B and V-C we also examine the generalization ability of the different models by adding 3 completely unseen classes, $D_{\text{unseen}} = \{\text{cheese, potato, lemon}\}$. For the experiments, the blocks are formed by $M = 10$ time-steps which correspond to $0.05s$ of measurements and in Sections V-B, V-C the prediction horizon is set to $H_b = 5$ blocks. For all the networks we used the Adam optimizer [34] with the hyperparameters learning rate (lr), weight decay (wd) and learning rate decay (gamma) set as listed in Table I.

### Table I: Hyperparameters used for the experiments.

| Model       | lr    | wd    | gamma |
|-------------|-------|-------|--------|
| RNN         | 1e-04 | 5e-04 | -      |
| LSTM        | 1e-04 | 5e-04 | -      |
| LSTM-c      | 1e-04 | 3e-04 | -      |
| LSTM-lr-c   | 1e-04 | 3e-04 | 0.5    |

From the results in Fig. 3 it can be seen that for short horizons, the LSTM networks have comparable results, while the RNN displays higher error. As the horizon increases, the performance of all the networks, except LSTM-lr-c, degrades to the same point. Before the prediction horizon reaches $t = 0.45s$, LSTM-c has only marginally better results than its simpler counterpart, showcasing that simply employing a learning curriculum is not enough to boost the predictive performance. Finally, throughout the experiment, LSTM-lr-c significantly outperforms all of the baselines, supporting that the combination of learning rate decay and curriculum training results in better performance that scales well with the prediction horizon.

Secondly, we report the average MSE during forward predictions on a test set for a prediction horizon of $H = 5$ blocks. For this purpose, we performed trials for 5 object classes that were also in the training set $D_{\text{seen}}$ and the additional 3 classes in $D_{\text{unseen}}$. We recorded two repetitions for three different values of $K_a$ amounting to a total of 30 trials with seen classes and 18 trials with unseen ones.

### Table II: Test performance while predicting relative displacements for 5 blocks in the future. "Seen classes" include unseen datasets but on objects that have been treated in training as opposed to the "Unseen Classes" that have never been encountered.

| Model       | Seen classes $(10^{-5} \text{ mm})$ | Unseen classes $(10^{-5} \text{ mm})$ | Total $(10^{-5} \text{ mm})$ |
|-------------|-----------------------------------|--------------------------------------|-------------------------------|
| RNN         | 2.08                              | 3.34                                 | 2.55                          |
| LSTM        | 2.26                              | 3.75                                 | 2.82                          |
| LSTM-c      | 2.30                              | 3.94                                 | 2.92                          |
| LSTM-lr-c   | 1.37                              | 2.29                                 | 1.72                          |

Table II shows the corresponding results for each model on seen and unseen classes, as well as the total MSE for both cases. It is evident that LSTM-lr-c is consistently better than the rest of the models and generalizes well to the unseen cases. It is interesting to observe that despite its poor scaling as the horizon grows, the RNN model shows slightly better results than the LSTM baselines in both datasets. This reinforces the results from the previous section concerning the training procedure and further indicates the usefulness of combining curriculum training with learning rate decay.
C. Robotic experiments

To evaluate the models’ performance within the controller, we executed a series of experiments with the 9 different object classes in $D_{\text{seen}}$ and $D_{\text{unseen}}$. In the experiments, we used a YuMi-IRB 14000 collaborative robot with an OptoForce 6-axis Force/Torque sensor mounted on its wrist. For every model, we executed 5 trials per class. Throughout the trials, we set $K_a = 0.003I_3$ and kept the same cost function. The trial ended successfully only when the knife had reached the cutting board. In any other case, e.g. if the execution time exceeded a minute, the trial was considered unsuccessful and the results were discarded. Since we have included sequences of free-space motion in the training data, we did not initialize the trials with the knife already in the object as in [12], [15], but directly above it with no further indications of the object’s location.

It should be noted that due to the robot’s hardware limitations, stiffer objects often cause the torque limit to be surpassed, leading the robot to shut down, which constituted a failure and the trial was repeated. This was especially evident while evaluating the RNN architecture on lemons as the closed-loop behavior did not exhibit the necessary sawing motion to break friction. This delayed the downwards progress, or simply caused a hardware failure, making it impossible to collect successful trials. For this reason, in Table III the associated results are marked with a double asterisk to denote that RNN trials were omitted when calculating the average scores.

In Table III we assess the online performance through the mean cost achieved by the models. For almost half of the classes, LSTM-lr-c again outperforms the rest of the baselines and achieves better mean costs. However, it is interesting to notice that LSTM-c, despite having the highest MSE during forward predictions, still manages to perform well, and more importantly, accomplishes the best scores in 2 out 3 unseen cases. Finally, even though RNN failed to complete a cut on lemons, it still has notable performance on several classes. This is partially due to the fact that the strategies it resulted in revolved mostly around slicing the object instead of sawing. However, because this behavior was more aggressive, it often led to torque limit violations.

In a successful cutting trial, it is straightforward to surmise that the main objective is downwards motion. Nevertheless, the sawing motion is related to the downward progress, as it enables it by breaking friction and minimizing the sheer force otherwise required. Consequently, apart from the mean cost, a crucial point of evaluation for the dynamics models is whether they lead the controller to infer useful strategies for each object class. For objects that are stiffer, fine-grained understanding of the dynamics should drive the strategy around sawing, while for the softer ones, it should deem it unnecessary. A qualitative demonstration of these emerging behaviors can be observed in Fig. 4a and Fig. 4b that depict the trajectories during trials on a soft (cake) and a stiff object (eggplant).

In the former case, any strategy is viable as there is no significant resistance from the material. LSTM that had the best cost for this class, results in minimum sawing, as it is redundant, and so is LSTM-lr-c despite it’s worse cost-wise performance. On the other hand, RNN, that had almost the same cost as LSTM, displays similar behaviour with the worst model for this class. In the latter case of the eggplant, it is substantially more difficult to cut through the object without sawing, because of its density and firmness. LSTM-lr-c demonstrates the most insightful behavior with

### Table III: Mean cost across trials. Object classes denoted with an asterisk belong to $D_{\text{unseen}}$. The results denoted with a double asterisk do not take into account the failed attempts.

|          | Cake | Zucchini | Cucumber | Banana | Pepper | Eggplant | Cheese* | Potato* | Lemon* | Avg.    |
|----------|------|----------|----------|--------|--------|----------|---------|---------|---------|---------|
| RNN      | 1.08 | 1.04     | 1.49     | 1.59   | 0.92   | 1.42     | 0.71    | 0.55    | N/A     | 1.13**  |
| LSTM     | 1.07 | 2.34     | 0.96     | 1.72   | 2.08   | 2.11     | 1.69    | 0.68    | 1.43    | 1.56    |
| LSTM-c   | 1.51 | 1.41     | 1.68     | 1.19   | 1.21   | 0.98     | 2.18    | 0.51    | 0.99    | 1.29    |
| LSTM-lr-c| 1.26 | 0.91     | 0.71     | 2.31   | 0.72   | 0.95     | 0.73    | 1.76    | 1.80    | 1.24    |
| Avg.     | 1.23 | 1.42     | 1.21     | 1.71   | 1.23   | 1.36     | 1.33    | 0.88    | 1.16**  |

Fig. 4: The sawing trajectories followed by the controller during two very distinct cutting cases. LSTM-lr-c leads the controller to insightful strategies depending on the dynamics of the class.
smooth sawing motions that led to the best cost. Similar behavior is adopted by LSTM-c that has the closest score, as opposed to LSTM and RNN that only employed low-magnitude sawing, which was not suitable for the dynamics of the class. In conclusion, even though LSTM-1r-c did not have the least cost for every class, it exhibited the most appropriately diverse techniques that were able to adapt efficiently to the dynamics encountered amongst the classes.

VI. DISCUSSION AND FUTURE WORK

In this work, we presented a data-driven framework for the contact-rich task of food-cutting. We showed that by carefully designing every step of the method, we can produce models that have consistently good predictive performance on known cases and generalize well to unseen ones. When evaluated with a predictive controller, the proposed approach achieved the best mean cost in 4 out of 9 object classes and displayed a better understanding of the dynamics as showcased by the strategies the controller adopted. In the future, we will explore avenues that allow adaption not only on different object sizes or classes, but on completely different and more complicated cases of cutting, such as objects with a large seed. To this end, we will further investigate the design choices as to seamlessly incorporate behaviors that could be otherwise generated by switching controllers or a high-level planner.

REFERENCES

[1] N. Hogan, “Impedance control: An approach to manipulation,” in 1984 American Control Conference, June 1984, pp. 304–313.

[2] J. D. Schutter and H. V. Brussel, “Compliant robot motion i. a formalism for specifying compliant motion tasks,” The International Journal of Robotics Research, vol. 7, no. 4, pp. 3–17, 1988.

[3] Y. Karayiannidis and Z. Doulgeri, “An adaptive law for slope identification and force position regulation using motion variables,” in Proceedings - IEEE International Conference on Robotics and Automation, vol. 2006, 06 2006, pp. 3538 – 3543.

[4] Y. Karayiannidis, C. Smith, F. E. V. Barrientos, P. Ogren, and D. Kragic, “An adaptive control approach for opening doors and drawers under uncertainties,” IEEE Transactions on Robotics, vol. 32, no. 1, pp. 161–175, Feb 2016.

[5] S. Chiaverini and L. Sciavicco, “The parallel approach to force/position control of robotic manipulators,” IEEE Transactions on Robotics and Automation, vol. 9, no. 4, pp. 361–373, Aug 1993.

[6] O. Kroemer, S. Niekum, and G. Konidaris, “A review of robot learning for manipulation: Challenges, representations, and algorithms,” ArXiv, vol. abs/1907.03146, 2019.

[7] A. Atkins and X. Xu, “Slicing of soft flexible solids with industrial applications,” International Journal of Mechanical Sciences, vol. 47, no. 4, pp. 479 – 492, 2005, a Special Issue in Honour of Professor Stephen R. Reid’s 60th Birthday.

[8] X. Mu, Y. Xue, and Y. Jia, “Robotic cutting: Mechanics and control of knife motion,” in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 3066–3072.

[9] P. Long, W. Khalil, and P. Martinet, “Force/velocity control for robotic cutting of soft materials,” in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 4716–4721.

[10] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable mpc for end-to-end planning and control,” in Proceedings of the 32Nd International Conference on Neural Information Processing Systems, ser. NIPS’18. USA: Curran Associates Inc., 2018, pp. 8299–8310.

[11] N. Mohajerian and S. L. Waslander, “Multi-Step Prediction of Dynamic Systems with Recurrent Neural Networks,” IEEE Transactions on Neural Networks and Learning Systems, vol. PP, pp. 1–14, 2018.

[12] I. Mitişioni, Y. Karayiannidis, J. A. Stork, and D. Kragić, “Data-driven model predictive control for the contact-rich task of food cutting,” 2019 IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2019.

[13] S. Jung and T. Hsia, “Adaptive force tracking impedance control of robot for cutting nonhomogeneous workpiece,” in Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), vol. 3, 1999, pp. 1800–1805 vol.3.

[14] G. Zeng and A. Hemami, “An adaptive control strategy for robotic cutting,” in Proceedings of International Conference on Robotics and Automation, vol. 1, 1997, pp. 22–27 vol.1.

[15] I. Lenz, R. A. Knepper, and A. Saxena, “Deepmpc: Learning deep latent features for model predictive control,” in Robotics: Science and Systems, 2015.

[16] M. Sharma, K. Zhang, and O. Kromer, “Learning Semantic Embedding Spaces for Slicing Vegetables,” 2019. [Online]. Available: http://arxiv.org/abs/1904.00303

[17] S. Rezaei-Sohdurti, D. Meger, and I. Sharif, “Learning the latent space of robot dynamics for cutting interaction inference,” vol. 3, 06 2020.

[18] P. Jamadagni and Y.-B. Jia, “Robotic cutting of solids based on fracture mechanics and fem,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 8252–8257.

[19] K. Kronander and A. Billard, “Learning compliant manipulation through kinesthetic and tactile human–robot interaction,” Haptics, IEEE Transactions on, vol. 7, pp. 367–380, 07 2014.

[20] B. Huang, M. Li, R. L. De Souza, J. J. Bryson, and A. Billard, “A modular approach to learning manipulation strategies from human demonstration,” Autonomous Robots, vol. 40, no. 5, pp. 903–927, Jun 2016.

[21] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, “Learning force-based manipulation of deformable objects from multiple demonstrations,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), May 2015, pp. 177–184.

[22] T. Tang, H.-C. Lin, and M. Tomizuka, “A learning-based framework for robot peg-hole-insertion,” in ASME 2015 Dynamic Systems and Control Conference, 10 2015, p. V002T27A002.

[23] Y. Fan, J. Luo, and M. Tomizuka, “A learning framework for high precision industrial assembly,” in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 811–817.

[24] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyy, J. A. Ojea, E. Solowjow, and S. Levine, “Residual reinforcement learning for robot control,” 2019 International Conference on Robotics and Automation (ICRA), pp. 6023–6028, 2019.

[25] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in Proceedings of the 26th Annual International Conference on Machine Learning, ser. ICML ’09. New York, NY, USA: ACM, 2009, pp. 41–48.

[26] F. Ebert, S. Dasari, A. X. Lee, S. Levine, and C. Finn, “Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning,” no. CoRL, pp. 1–12, 2018. [Online]. Available: http://arxiv.org/abs/1810.03643

[27] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, “Neural Task Programming: Learning to Generalize Across Hierarchical Tasks,” 2017. [Online]. Available: http://arxiv.org/abs/1710.01813

[28] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, “Sequence level training with recurrent neural networks,” in 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings, 2016.

[29] S. S. Sahoo, C. H. Lampert, and G. Martius, “Learning equations for extrapolation and control,” 2018.

[30] R. Jonschkowski, R. Hafner, J. Scholz, and M. Riedmiller, “PVMs: Position-Velocity Encoders for Unsupervised Learning of Structured State Representations,” 2017. [Online]. Available: http://arxiv.org/abs/1705.09805

[31] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” Journal of Machine Learning Research, vol. 9, no. Nov, pp. 2579–2605, 2008.

[32] J. Jiang, agraL Gülelhe, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” ArXiv, vol. abs/1412.3555, 2014.

[33] A. Rao, “A survey of numerical methods for optimal control,” Advances in the Astronautical Sciences, vol. 135, 01 2010.

[34] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” International Conference on Learning Representations, 12 2014.