Active Learning of the Cutting of Cooking Ingredients using Simulation with Object Splitting

Nahomi Ikegami[1], Solvi Arnold[1], Kotaro Nagahama[1] and Kimitoshi Yamazaki[1]

***Abstract*— Food preparation is hard to automate on general-purpose robot hardware. This work focuses on acquisition of cutting manipulations for food preparation. We construct a physical simulation environment for virtually performing cutting operations on food ingredients, including routines for splitting food object models at arbitrary positions, and design a compact manipulation description format. Using the simulation environment, we generate data for training a neural network architecture (EMD-net) to predict the state transformation induced by a given cutting manipulation and evaluate the network's performance. We also perform automatic generation of cutting manipulations for obtaining a given goal state from a given initial state, and evaluate the obtained manipulations by performing them in the physical simulation environment.**

I. INTRODUCTION

The cutting, peeling, and mixing of ingredients are essential elements of food preparation behaviour. Automating this work can be expected to yield reductions in labour, from large-scale settings such as food processing facilities, to medium-scale settings such as hospitals, to small-scale settings such as regular households. In large-scale settings each individual processing step can be assigned to a large specialized machine, for highly efficient production. However, medium and small-scale settings necessitate a single modestly sized machine capable of producing a variety of foods in small amounts each, in order to flexibly respond to demands. Recent years have seen increasing efforts to automate food preparation, including from new venture companies.

The goal of the present work is to efficiently acquire the manipulation abilities necessary for food preparation using a virtual environment. We consider preparation steps such as cutting, peeling, baking, etc. Such manipulations change the appearance of the ingredients. Hence in order to automate food preparation, we need functionality for grasping how ingredients will change when manipulated in a given way with a given tool. In this work, we realise this goal using an EMD network originally proposed for similar purposes.

Recent years have seen substantial progress in data-driven machine learning. Approaches using Neural Networks (NNs) in particular have produced impressive results. Much research is being performed on NN-based acquisition of robot behaviour. Such approaches generally require large amounts of training data, but the food preparation work considered here produces irreversible changes in the ingredients. Consequently, gathering sufficient data in a real-world setup is highly impractical. Hence the authors focus on data collection and system evaluation by means of physical simulation. We gather data by performing manipulations virtually in simulation, and recording the scene from a virtual camera. Furthermore, when we let the trained system generate tool manipulations to produce a given post-manipulation state, and evaluate the generated manipulations by performing them in the virtual environment.

The contributions of this work are as follows:

- We realised a mechanism for predicting the changes in objects subjected to cutting manipulations.
- Using the above, we generate manipulations for producing a given manipulation outcome.
- We propose simulation routines for modeling physical modifications involving cutting, making it possible to perform trial-and-error cutting work in a virtual environment. We automate data generation and system evaluation using these routines.

The paper is structured as follows. Section 2 discusses related work. Section 3 explains the problem setting and the structure of the proposed active learning system. Section 4 outlines the EMD-net at the core of the system's learning ability, and explains how it is applied here. Section 5 details our usage of simulation, and how it connects with the other parts of the system. Section 6 presents our experimental results. Section 7 concludes the paper.

II. RELATED WORK

A number of studies have addressed recognition of cooking action from video and images. Aboubakr et al. [1] proposed a method to identify the position of ingredients from video. They made it possible to recognize the state of the ingredients. Monteiro et al. [2] constructed a deep neural network to classify cooking actions from RGB images captured by a fixed camera. Yamakata et al. [3] proposed a method for ingredient recognition that divides cooking work into two parts: preparation phase and cooking phase. The former phase included peeling and cutting.

Some studies have addressed motion recognition using multi-modal sensor information. Hashimoto et al. [4] focused on a cutting process and report comparative results for various combinations of sensors. In addition to a ceiling camera, they showed that it is useful to install force sensors and sound sensors on the cutting board. By combining these three sensor types, they achieved a success rate of over 92% for distinguishing 9 types of vegetables. Inoue et al. [5]

[1]Nahomi Ikegami, Solvi Arnold, Kotaro Nagahama, and Kimitoshi Yamazaki are with AIS Lab., Faculty of Engineering, Shinshu University, 410, Mech. Sys. Eng. Buidling, 4-17-1, Wakasato, Nagano, Nagano, Japan {19w4004f, s_arnold, nagahama, kyamazaki}@shinshu-u.ac.jp

proposed a cooking assistance system for beginners. They proposed a classification method exploiting operator gaze and report high accuracy.

The studies introduced above are useful for applications where the computer-interpreted outcomes benefit human users. However, for the purpose of automation, some additional technological elements are required. For instance, generating robot motions requires clarification of the positional relationships between tools and ingredients.

Progress is also being reported in studies on robotic cooking on basis of recognition outcomes. Some studies target collaborative work with humans. For example, Sugiura et al. [6] achieved cooperative cooking with small mobile robots that are instructed using AR markers. Fukuda et al. [7] built a cooking robot system that supports the user with voice and gestures. Other studies addressed stand-alone robotic cooking. For example, Yang et al. [8] obtained knowledge representations for object manipulation from videos on the Internet. Watanabe et al. [9] present a unified system of life-size humanoid robots. Combining recognition of tools and ingredients with basic operation functions such as cutting, peeling and moving, their robot was able to autonomously prepare a salad. Amaro et al. [10] proposed a motion classification method that takes into account individual differences, and proposed a method for teaching motion to robots. They demonstrate pancake and sandwich preparation by a humanoid robot following instruction by multiple people.

In the studies discussed above, food processing was realised at the granularity of cutting, baking, mixing and the like. However, as discussed in section 1, practical application requires a finer level of granularity than this. For instance, substantial knowledge is needed for even just cutting ingredients with a knife. There have been studies focusing on such tasks. Mu et al.[11] succeeded at cutting several types of vegetables using a 2 DoF manipulator. Based on data obtained from force sensors and torque sensors installed on the manipulator, cutting trajectories were adjusted online. However, this study did not focus on determination of cutting positions or state recognition after cutting. Therefore, applicability to automated food processing is limited.

III. Challenges and approach

Food processing generally involves irreversible modification of the ingredients. For learning-based approaches, such modifications present a tough issue. One hurdle is the enormous burden of collecting training data. Once processed, ingredients cannot be re-used, because their original state (shape, color and so on) cannot be restored. Therefore, producing sufficient training data would require massive quantities of the ingredient items.

An alternative approach is to collect training data using a virtual environment that can simulate ingredient processing. This requires replication, in a physics simulation, of the modifications induced by cooking utensils in food ingredients.

However, programming environments currently used in the field of robotics generally lack functionality for simulating modifications such as object splitting or color changes, let alone deformations. To the best of our knowledge, no existing environment meets the demands posed by the present study. For instance, we develop a cutting motion generator, and need to examine its performance. Specifically, we need to process an ingredient item on basis of the output of the generator, and evaluate the result. In such a case, we need to be able to set the position at which the ingredient is split in online fashion.

For the case of a cutting task, we need functionality for predicting the state that would result from a given cutting operation. Such functionality makes it possible to determine what operations should be performed in what order. For example, when we need to cut an ingredient item into several slices, we can determine in advance the order and positions of the cuts to be made.

Considering the above, we take the following approach:

- We propose a novel method for representing object splitting in physical simulation, and implement functionality for reproducing the cutting of ingredients.
- Incorporating the above functionality and a learning algorithm, we construct a unified learning system that integrates data generation in a virtual environment, training using this data, and evaluation of the outcomes in the virtual environment.

The proposed method consolidates all processes from data generation to evaluation into a virtual environment, reducing the data collection burden.

Fig. 1 shows the overall structure of the proposed method. The system consists of a physics simulator and a motion generator. During the training process, a dataset produced by the physics simulator provides the input for the motion generator.

After the training process, we let the motion generator generate motions on basis of image pairs comprised of an initial state and a target state. The generated motions are sent to the physics simulator for execution and outcome evaluation. When dealing with a task comprising multiple steps, the post-execution state is sent back to the motion generator as the initial state for the next step.

For generating a dataset with the physics simulator, we construct a virtual kitchen by placing an ingredient item, a knife, and a cutting board in the simulation environment. A virtual camera is also installed in the simulation environment, which captures an image of the scene before and after each cutting operation. By repeating the process of cutting and image capture numerous times with various settings for the cutting operation's parameters, we collect a training dataset.

Using the dataset produced in the physical simulation environment, the manipulation generator is trained for both manipulation generation and manipulation outcome prediction. The trained generator infers manipulations given images of the pre- and post-manipulation states. Simultaneously, it also outputs an image representing its prediction of the post-manipulation state for the inferred manipulation. Finally, the generated manipulation is sent to the simulator, where evaluation is performed. We evaluate manipulations by re-

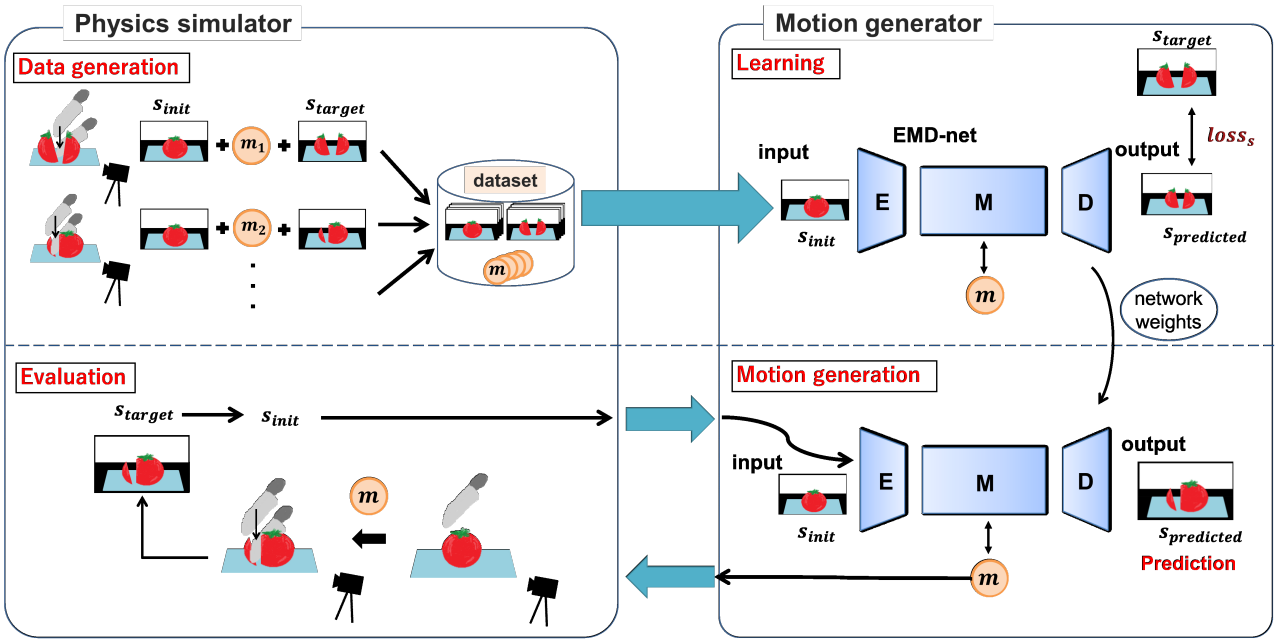

Fig. 1. Overview of the proposed active learning system

constructing the pre-manipulation state in the simulator and performing the generated manipulation on it.

IV. EMD-NET FOR INGREDIENT MANIPULATION

A. EMD-net

The EMD-net is a neural network architecture for predicting manipulation outcomes and planning manipulations [12] The network is composed of 3 modules. Convolutional encoder module E maps (encodes) state s_i (an image representing the task space at time i) to a latent representation c_i:

$$E(s_i) = c_i \tag{1}$$

Fully connected manipulation module M maps latent representation c_i and manipulation m_i to a prediction $\hat{c}_{i+1}$ of c_{i+1}, a latent representation of the state s_{i+1} that results from applying manipulation m_i to s_i :

$$M(c_i, m_i) = \hat{c}_{i+1} \tag{2}$$

Up-convolutional decoder module D maps (decodes) latent representation $\hat{c}_i$ to an approximation $\hat{s}_i$ of state s_i:

$$D(\hat{c}_i) = \hat{s}_i \tag{3}$$

All convolution kernels have size (3 × 3 × 3). We avoid pooling, as it discards relevant spatial information. Resolution on the spatial dimensions is reduced by means of strided convolution (stride size 2) instead. The 7 layers of E have 3, 32, 32, 64, 128, 256 and 512 feature maps. Feature map counts in D mirror E. M employs residual connectivity, with each layer copying the activation values corresponding to the latent representation from the previous layer before forward propagation. Manipulation inputs are duplicated in every layer of M, feeding the manipulation input identically at every layer. Inclusion of residual connectivity and repetition of the manipulation inputs were found to improve network performance in comparative experiments on a different task [12]. We combine the modules into the composite network $D(M(E(s_0), m_0))$.

After training, we can use the EMD-net to predict the outcome of a manipulation by feeding the initial state and the manipulation as input into the EMD-net and performing forward propagation.

The EMD-net also allows us to generate manipulations, using the concept of planning by backpropagation. When planning a manipulation we assume to be given the initial state s_{init} and target state s_{target}. We then generate a manipulation for producing the latter from the former as follows:

1) Let m be a randomly initialized manipulation.
2) Obtain the expected outcome state $s_{predicted}$ for m by forward propagation with s_{init} and m as inputs.
3) Compute the MSE loss over $s_{predicted}$ and s_{target}.
4) If the loss has not improved for 25 iterations or 100 iterations have passed, return the best manipulation evaluated so far.
5) Obtain loss gradients w.r.t. the manipulation inputs by means of backpropagation of the loss through M.
6) Adjust m in accordance with these gradients using the iRprop- update rule [13], and return to step 2.

We run 10 parallel instances of this process on GPU, and select the plan with the lowest remaining loss.

B. Application of EMD-net to food preparation

Humans decide how to cut ingredients based on expectations about how a given cutting operation will change the

state of the ingredient. In order to select cutting operations on basis of their predicted outcomes, we need the following functionalities.

- The ability to predict how a given cutting operation will modify the state of the ingredient.
- The ability to infer a cutting operation from its preceding and resulting states.

We require the ability to predict state changes in addition to the ability of operation inference for the purpose of verifying that a given inferred operation can indeed transition the ingredient into the goal state. The EMD-net introduced in the previous section provides both of these functionalities, and hence we adopt it here.

For both state transition prediction as well as operation inference using the EMD-net, the set of parameters we choose for representing operations is of crucial importance. The manipulation network uses valuations over these parameters to transform the latent representation of the pre-operation state into a latent representation of the predicted post-operation state. As such, the chosen representation should fully define the operation. However, using a large number of parameters will increase training times, as well as the computational cost of operation inference. Finally, it is also important that inferred operations can be properly translated into executable motion instructions for the robot.

In the present work we define the network's input and output as follows. The initial state and goal state are given as 4-channel images combining RGB and D (depth) images. The manipulation parameters are chosen to describe cutting operations, defining the orientation of the knife and its position relative to the cutting board.

We train and evaluate the EMD-net using the following error metrics.

- $loss_s$: the prediction loss, i.e. the MSE over the predicted state $D(M(E(s_{init}), m))$ and the actual post-manipulation state s_{target}.
- $loss_c$: latent representation loss. This loss quantifies the difference between $E(s_{target})$ and $M(E(s_{init}), m)$.

$$loss_c = \frac{\sum (M(E(s_i), m_i) - E(s_{i+1}))^2}{size_c} \quad (4)$$

  With $size_c$ being the size of the latent representation. This loss serves to enable multi-step prediction and planning [12], which is not addressed in the present paper.
- $loss_r$: reconstruction loss, i.e. the MSE over $D(E(s))$ and s, for $s \in \{s_{init}, s_{target}\}$. Training with this loss helps in acquiring the ability to map full state representations to latent representation and vice-versa.

V. Physical Simulation

A. Realisation of object splitting

To simulate object splitting manipulations, we adopt the Gazebo simulator [14], which is commonly used in the field of robotics. Because Gazebo is a ROS plugin, it is comparatively easy to run simulations with models from the ROS environment, as well as self-made models. Within Gazebo, we construct an environment mimicking a real food preparation setting, as shown in Fig. 2.

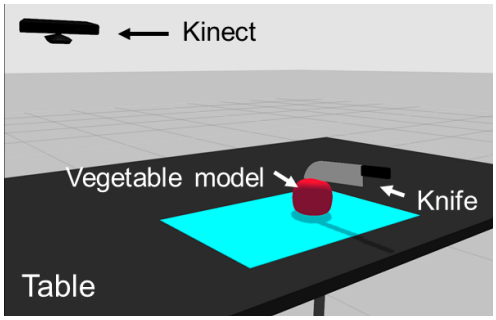

Fig. 2. Virtual kitchen environment for Gazebo physics simulator

However, Gazebo has no functionality for splitting models. Here we circumvent this limitation as follows. We construct models out of separate blocks, and move these blocks apart when a cutting operation is performed on the model. In method 1, shown in the left panel of Fig. 3, we define the model as a set of pre-cut blocks placed seamlessly against one another. For example, when cutting a cucumber into three parts, we construct the cucumber model by placing the three cylindrical parts to be obtained after cutting seamlessly against one another, thereby obtaining a model looking indistinguishable from a cucumber constructed as a single block. Then the cutting position is chosen from among the splits in the model. Within Gazebo, each part has a separate "link". By assigning coordinates to the link, the part's position can be controlled. By so moving the links of the model parts to the side of the knife to which the knife is slid after the cut, by the same distance the knife is slid, we obtain proper splitting of the food item.

Gazebo can read models created in the STL format, so it is possible to construct arbitrary post-manipulation states. However, this in effect determines the possible cutting positions and outcomes before simulation. Therefore, we devised an alternative method (method 2) for handling cutting in simulation, shown in the right panel of Fig. 3. Model definitions in Gazebo are sdf files in xml format. We start by creating cubes and cylinders with sides of length 0.01m. When launching the simulation, we parse the xml code in the sdf file and rewrite the model's size and color parameters, after which the resulting file is read in by Gazebo. The position at which the model is split is calculated on basis of randomly chosen model placement and cutting trajectory parameters.

By adjusting the scales of the parts comprising the model, leaving a seam for the knife, we create a model that is split at the required position, without imposing constraints on the positions of the cuts. Finally we move the knife as if performing the cutting operation, thereby causing the model parts to separate naturally in accordance with the movement of the knife (Fig. 3, right panel). Using this method we can simulate the cutting and separating of objects at arbitrary

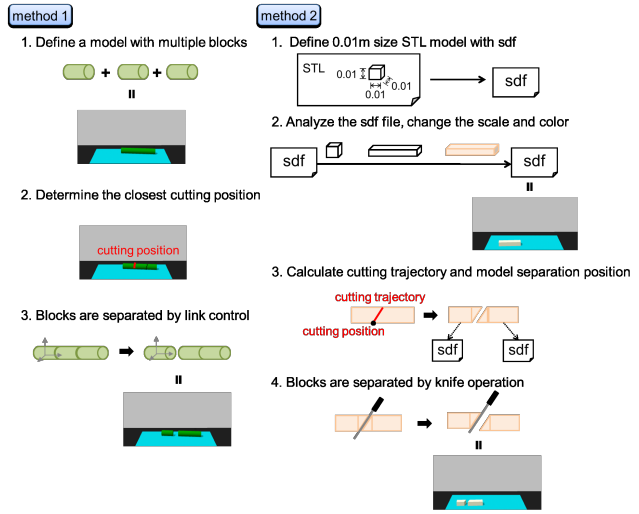

Fig. 3. Methods for object splitting in physics simulation

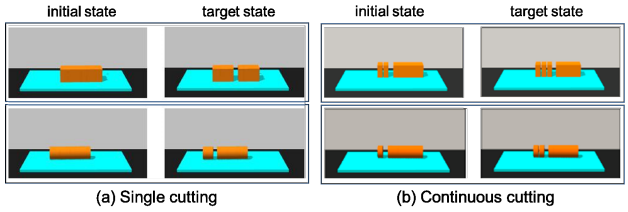

Fig. 4. Examples of training data: (a) cuboid and cylindrical model for single cuts, (b) cuboid and cylindrical model for sequential cuts.

positions, because the cutting position is not fixed.

B. Training data collection

We place a table, cutting board, and imitation food ingredients in the virtual environment, and simulate cutting operations. We record the scene from a virtual RGBD camera aimed at the cutting board, storing the RGB and depth images obtained for the pre- and post-manipulation states together as one datum. These images show scene content aside from the ingredients as well, but we perform no background removal or other such processing on the images, instead letting the EMD-net handle prediction of the full scene visual. We vary the color and shape of the ingredient models, and the position and angle at which they are placed on the cutting board. By obtaining pre- and post-manipulation states for many variations, we obtain diverse data. Furthermore, when generating batches during training, we perform data augmentation by randomly varying image contrast.

VI. Experiments

A. Settings

We conducted experiments on cutting pseudo-vegetables. When humans cut food ingredients, they attend closely to the position and the direction of the knife as it cuts into the ingredient. The way the shape of the ingredient changes varies depending on the knife's movement during cutting. Therefore, the manipulation parameters should include the cutting position, knife orientation, and the knife's motion after making the cut. In this experiment, we consider the following cutting motion: the knife is perpendicularly lowered to cut the ingredient, and is then moved sideways to separate the parts. The manipulation parameters for the knife motion are as follows: cutting position (m_x, m_y, m_z), knife orientation $(m_\theta, m_\phi, m_\psi)$, and knife motion after the cut (longitudinal component m_a and lateral component m_b).

RGB and depth images were rendered at a resolution of 128x256. The collected data was randomly divided into three sets: 80% was used for training, 10% was used for testing, and the remaining 10% was used for validation. The batch size for training was 16.

The arrangement of ingredients and tools in our simulator was as follows. A cutting board of size 0.24 m $\times$ 0.382 m $\times$ 0.001 m was located on a tableof height 0.774 m. A shape model representing a vegetable was placed on the cutting board. A camera to capture RGBD images was located 1.15 m away from the center of the cutting board, at a height of 1.15 m, tilt downward by 0.27 rad. The length of the blade of the knife was 0.13 m and the point of first contact with the vegetable models was at 0.03 m from the tip.

The mass and the friction of the models do not match those of actual ingredients and tools exactly. The cutting board was fixed to the table so as not to slip when the vegetables and the knife move around on top of it.

B. Outcome prediction for cutting operations

We evaluated the EMD-net's ability to predict the state transformations induced by cutting manipulations as follows. Using a dataset of 2000 items of data from the cuboid and cylindrical models seen in Fig. 4(a), we trained the EMD-net for about 4 days.

Fig. 5 shows the evolution of the loss measures w.r.t. test and training data over the course of training. The vertical and horizontal axes correspond to the loss magnitude and training step, respectively. Losses for training data are plotted every 10 steps, and those for test data every 500 steps. As can be seen in Fig.5, $loss_s$ converges to a satisfactory value of about 0.001 for training data. However, when we compare the $loss_s$ and $loss_c$ values obtained for training data with those obtained for test data, we observe that from some point onwards, the losses for test data stop decreasing and instead start to increase, resulting in a significant difference with the training losses at convergence. This suggests overfitting. Overfitting should be reducible by increasing the size of the dataset.

During training, we evaluate the network's performance on the validation set once every 10,000 training steps. For further evaluation, we use the network that obtained the best validation score. In this experiment the best validation score was obtained at step 250,000. Examples of state predictions generated by this network are shown in Fig. 6. From left to right, each row shows an initial (pre-manipulation) state, a goal (post-manipulation) state, and the corresponding prediction generated by the network. Each state consists of an RGB image (top part) and a depth image (bottom part). The top

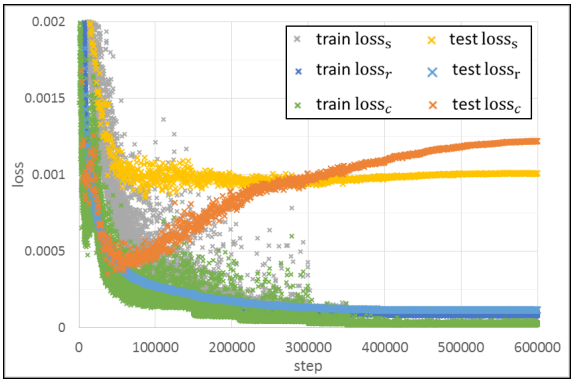

Fig. 5. Evolution of loss values during network training (single cuts).

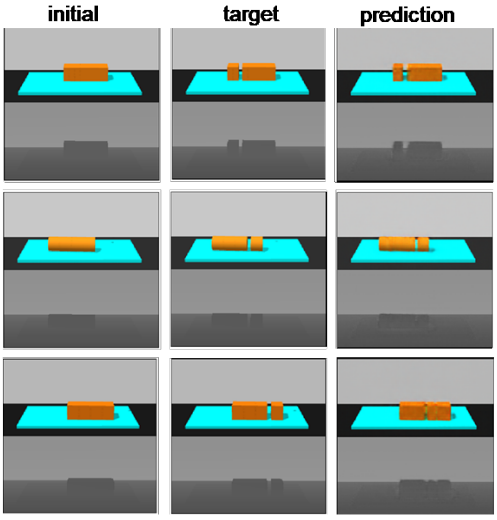

Fig. 6. Prediction results (RGB and depth) from the network at training step 250,000. **Top :** cuboid model, **middle :** cylinder model, **bottom :** failure case.

and center rows show close similarity between goal state and predicted state in both the RGB and depth image, for both vegetable models. However, we also observe cases where the position of the cut in the predicted state differs from that in the goal state. An example is shown in the bottom row of Fig. 6. Such cases occur with test data only; for training data not a single case of substantial discrepancy between goal state and prediction was observed. This too is strongly suggestive of overfitting. In a sampling of 16 predictions on test data, we found that 12 matched the goal state, thus obtaining a success rate of 75%.

C. Prediction for sequential cutting operations

The cutting operations seen in real-world food preparation are typically performed in sequence, i.e. ingredients are cut a number of times. Consequently, we need to be able to predict the state transformations produced by cutting operations performed in sequence. However, this requires prediction for cuts performed in states where the ingredient is already in a cut state. Therefore, we augment the dataset discussed above with 1000 items of new data (covering both the cuboid and cylindrical model, Fig. 4(b)) wherein additional cuts are performed from states where the ingredient has already been cut some number of times, and again run our evaluation experiment. As before, we train the EMD-net for about 4 days. The evolution of the loss values over the training process is shown in Fig. 7. For training data, $loss_s$ converges to a satisfactory value of 0.0008. Comparing the losses to those obtained in the previous section (Fig. 5), we observe that the gap between training loss and test loss has decreased for both $loss_s$ and $loss_c$, and that they converge to smaller final values. This likely reflects a reduction in overfitting due to the increased dataset size.

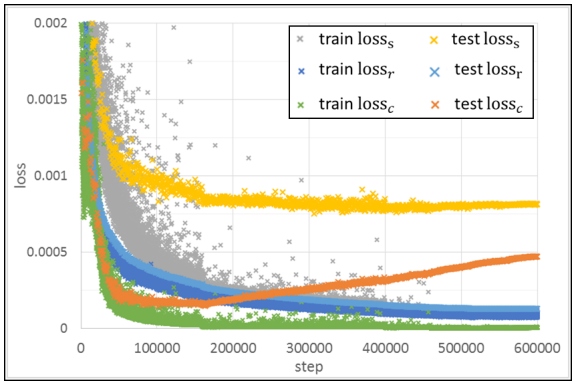

Fig. 7. Evolution of loss values during network training (sequential cuts).

The best validation score was obtained at iteration 410,000. Predictions generated by the corresponding network are shown in Fig. 8. As illustrated in the top row of Fig. 8, state transformations for data as used in the previous section (Fig. 4(a)) are predicted correctly. Furthermore, for the type of data newly added to the dataset, too, correct predictions are obtained. This is illustrated in the bottom row of Fig. 8, where we see no difference of substance between the goal state and the prediction. For this experiment, we obtained 11 correct predictions for a sampling of 16 examples from the test set, yielding a success rate of 69%. This establishes that it is possible to predict the outcome of performing an additional cut in a state where some number of cuts has previously been performed. This makes it possible to predict outcomes of operation sequences by performing prediction one operation at a time, and suggests the possibility of extending the proposed method so as to handle prediction for operation sequences.

D. Generation of cutting manipulations

Next we conducted an experiment to evaluate cutting motions generated by the EMD-net. For training, a dataset was prepared containing 7850 training examples of pseudo-vegetable, as shown in Fig. 9(a), and 2100 training examples of simple geometrical shapes, as shown in Fig.9(b). The network was trained for 1,330,000 steps (again taking approximately four days). Training in this experiment minimized $loss_s$ and $loss_c$ only. The network from step 1,010,000 was selected for evaluation. This evaluation employed 100 examples of test data. Fig. 10 shows snapshots of a cutting

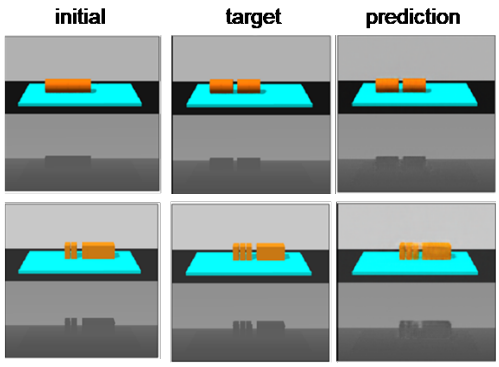

Fig. 8. Prediction results (RGB and depth) from the network at training step 410,000. **Top** : base dataset, **bottom** : expanded dataset.

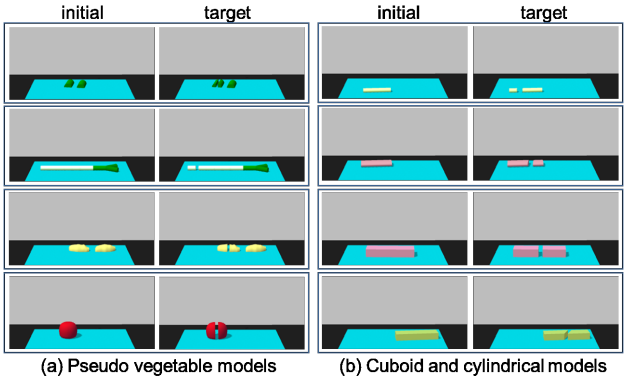

Fig. 9. Examples of (a) pseudo-vegetable models and (b) simple geometrical shapes used in the motion generation experiment.

operation, and Fig. 11 shows example results of performing motions generated by the EMD-net in simulation.

In these figures we can observe that physics simulation and state prediction produced appropriate results. We also observed that the cutting motions generated by the EMD-net can successfully realize given goal states, as shown in the left side of Fig.11. However, in some cases, the manipulation parameter values generated by the network differed substantially from the ground truth. An example is shown in right side of the figure. These results show that the proposed framework, consisting of a physics simulator and a motion generator, work as intended, but also indicate that there is room for improvement, especially in motion generation.

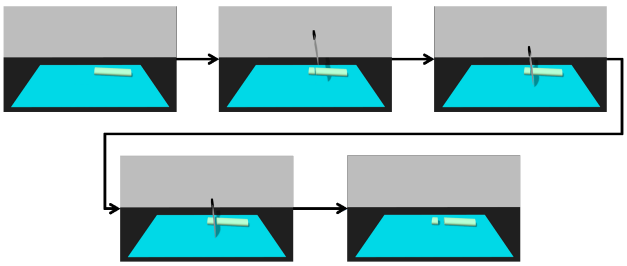

Fig. 10. Snapshots of a simulated cutting motion.

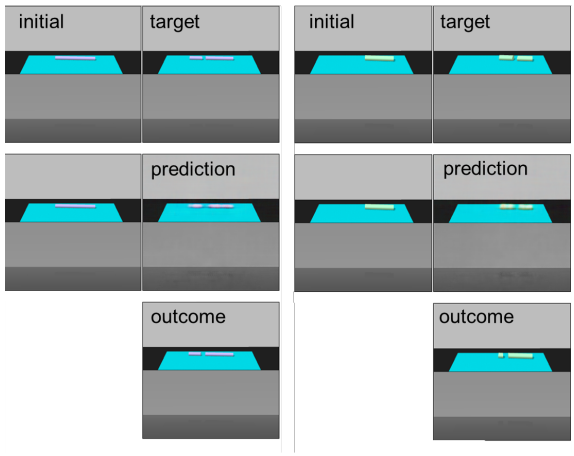

Fig. 11. Results of manipulation generation. **Left** : successful example, **right** : failure example.

TABLE I
ERROR MEASUREMENTS FOR CUTTING POSITIONS GENERATED BY EMD-NET

	m_x	m_y	m_z
minimum	0.00678	0.00335	0.00274
maximum	0.14808	0.06302	0.051379
average	0.09048	0.02728	0.02398

Table I shows the minimum, maximum, and average values of the error on the cutting position parameter values generated by the EMD-net. We observe that the largest deviations occur on the x axis, where the error is 0.14m at maximum and 0.09m on average. In the context of our cutting task, a deviation of 0.09m is a large error. Furthermore, the minimum error is in the order of a few mm. This suggests that the system may struggle when it comes to cutting ingredients into thin slices. In addition, there were cases where simulation of the generated motion was not possible, such as cases where the initial knife position specified in the motion intersects with the cutting board.

One cause of error in the generated manipulation parameters is that the reference coordinate system is tilted downward. Due to this tilt, the valid range of some parameter values differs depending on the cutting position in the z direction. Possible countermeasures are adjustment of these values in accordance with the z position, or reconsideration of the manipulation parameter specification. Performance may also be improved by further increasing example variety and the size of the dataset.

VII. CONCLUSION

In this study, we proposed a method to acquire manipulation abilities for splitting objects into pieces. In order to reduce the burden of collecting data in the real world, we adopted an approach using a physics simulator. Focusing on the task of cutting ingredients with a knife, we applied a method for predicting the state resulting from a given cut, and for generating cutting manipulations to achieve a given result state. With regard to the physics simulation, we proposed an

implementation method for adjusting object shape change on the spot. Combining these functionalities, we presented a framework that allows for training, evaluation, and manipulation generation, without human intervention.

In our evaluation experiments, food ingredients were represented using simple shapes such as cylinders and rectangular cuboids, and we implemented a physics simulation that separates the object into two by means of plane cutting. Training data was collected using this simulation, and network training for state prediction and manipulation generation was performed. For the purpose of network training, we examined how to appropriately parametrize the cutting manipulation. We showed that our method can predict outcome states with a certain degree of accuracy. We also experimented with manipulation generation and execution of these manipulations in the physics simulator. Although there is still room for improvement in the accuracy of the generated manipulations, we confirmed the proposed framework's suitability for motion acquisition. In addition, areas in need of further improved were identified through these experiments.

Future work includes accuracy improvement for manipulation generation. This will require re-examination of the choice of manipulation parameters. We also intend to increase the amount and variety of training data. Another important extension is to increase model complexity to more closely approximate real-world objects.

ACKNOWLEDGMENT

This work is partly supported by NEDO.

REFERENCES

[1] Nachwa Aboubakr, Remi Ronfard, and James Crowley, "Recognition and localization of food in cooking videos", CEA/MADiMa '18 Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management, pp. 21-24, 2018.

[2] J. Monteiro, R. Granada, R. C. Barros and F. Meneguzzi, "Deep neural networks for kitchen activity recognition", International Joint Conference on Neural Networks, pp. 2048-2055, 2017.

[3] Yuko Yamakata, Koh Kakusho, and Michihiko Minoh, "Object Recognition based on Object's Identity for Cooking Recognition Task", ISM 2010, 12th IEEE International Symposium on Multimedia, Taichung, Taiwan, 2010.

[4] Atsushi Hashimoto, Jin Inoue, Kazuaki Nakamura, Takuya Funatomi, Mayumi Ueda, Yoko Yamakata, and Michihiko Minoh, "Recognizing Ingredients at Cutting Process by Integrating Multimodal Features", CEA '12 Proceedings of the ACM multimedia 2012 workshop on Multimedia for cooking and eating activities, pp. 13-18, 2012.

[5] Hiroya Inoue, Takatsugu Hirayama, Keisuke Doman, Yasutomo Kawanishi, Ichiro Ide, Daisuke Deguchi, and Hitoshi Murase, "A classification method of cooking operations based on eye movement patterns", ETRA '16 Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, pp. 205-208, 2016.

[6] Yuta Sugiura, Daisuke Sakamoto, Anusha Withana, Masahiko Inami, and Takeo Igarashi, "Cooking with Robots: Designing a Household System Working in Open Environments", CHI2010, Cooking, Classrooms, and Craft, DOI:10.1145/1753326.1753693, 2010.

[7] Tsukasa Fukuda, Yasushi Nakauchi, Katsunori Noguchi, and Takashi Matsubara, "Sequential Human Behavior Recognition for Cooking-Support Robots", JRM vol.17 No.6, pp.717-724, 2005.

[8] Yezhou Yang, Yi Li, Cornelia Ferüller, and Yiannis Aloimonos, "Robot Learning Manipulation Action Plans by " Watching " Unconstrained Videos from the World Wide Web", AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 3686-3692, 2015.

[9] Yoshiaki Watanabe, Kotaro Nagahama, Kimitoshi Yamazaki, and Masayuki Inaba, "Cooking Behavior with Handling General Cooking Tools based on a System Integration for a Life-sized Humanoid Robot", Paladyn, Journal of Behavioral Robotics, Volume 4, Issue 2, pp. 63–72, ISSN (Print) 2081-4836, 2013.

[10] Karinne Ramirez-Amaro, Emmanuel Dean-Leon, and Gordon Cheng, "Robust Semantic Representations for Inferring Human Co-manipulation Activities even with Different Demonstration Style", 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 1141-1146, 2015.

[11] Xiaoqian Mu, Yuechuan Xue, and Yan-Bin Jia, "Robotic Cutting: Mechanics and Control of Knife Motion",2019 International Conference on Robotics and Automation (ICRA), DOI: 10.1109/ICRA.2019.8793880, 2019.

[12] Solvi Arnold, Kimitoshi Yamazaki, "Fast and Flexible Multi-Step Cloth Manipulation Planning Using an Encode-Manipulate-Decode Network (EM*D Net)," Frontiers in Neurorobotics, Vol. 13, DOI: 10.3389/fnbot.2019.00022, May 2019.

[13] Christian Igel and Michael Hüsken, "Improving the Rprop Learning Algorithm," in Proceedings of the Second International Symposium on Neural Computation, NC'2000, pp. 115-121, 2000.

[14] N. Koenig, and A.Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator ", 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vol. 4, DOI: 10.1109/IROS.2004.1389727, 2004.