2022 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics (AIM)

Food Arrangement Framework for Cooking Robots

Hoshito Nagahama[1], Ixchel G. Ramirez-Alpizar[2,1] and Kensuke Harada[1,2]

***Abstract*— We propose a food arrangement framework for a robot to automatically serve meals. We start from the premise that anybody *knows* how to arrange food. Based on this, we use a Convolutional Neural Network (CNN) to evaluate how good a food arrangement is. The CNN is trained using a dataset gathered through Amazon Mechanical Turk (AMT), where people are asked to choose the best food arrangement between a pair of pictures. The food arrangement and rearrangement is done entirely virtual through image processing. The initial food placement is random and evaluated by the CNN. If this evaluation is under a given threshold, the position of some ingredients will be changed according to the rearrangement algorithm we developed. For this algorithm we tested two different strategies for finding a relocation together with two different approaches for deciding which food to relocate. After the rearrangement is done, the CNN will evaluate again the food arrangement. The previous steps will be repeated until the food arrangement evaluation is beyond the given threshold. The resulting arrangement will be given to the robot for its actual execution. We evaluated our framework using two different sets of meals. We demonstrate that a UR3 robot is capable of serving a steak meal using a spatula-like end-effector.**

I. Introduction

Nowadays, the existence of robots in several scenarios is becoming familiar in our society, e. g., communication robots, cleaning robots, etc. In recent years, specialized cooking robots such as OctoChef [1] and the Okonomiyaki cooking robot [2] have been developed for business purposes. Therefore, these robots are programmed to prepare a single kind of meal. Contrary to this, our aim is to develop a cooking robot able to cook and serve a broad range of meals for household use. A robot that could be able to prepare and put on a plate any kind of meal and bring it to the dinning table. Serving food on a plate might seem trivial, nevertheless an important part of serving a meal is its presentation. Michel et al. [3] showed that visual presentation influences the perceived taste of a salad for three different presentations using the exact same ingredients, the salad with the better taste was that with an artistic presentation. Most people know how to *nicely* arrange food on a plate. Although *nicely* could be subjective, there is common sense about this. Why? Because we unconsciously learn this through observation of previous meals. However, this is a non trivial task for a robot. Having this in mind, we focus on how to transfer this common knowledge that a human has about how to arrange food on a plate to a robot.

[1]Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama-cho, Toyonaka, 560-8531, Japan. harada@sys.es.osaka-u.ac.jp

[2]Automation Research Team, Industrial Cyber-Physical Systems Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan. ixchel-ramirezalpizar@aist.go.jp

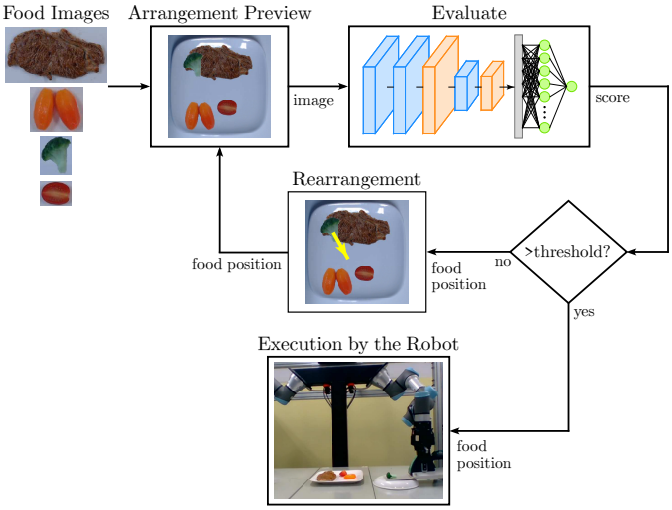


Fig. 1: Outline of the proposed food arrangement framework.

In this paper, we propose a food arrangement framework for a robot to carry out the final arrangement of food in plates, as shown in Fig. 1. We trained a Convolutional Neural Network (CNN) using a dataset gathered through Amazon Mechanical Turk (AMT), where people are asked to choose the best food arrangement between a pair of pictures. Like this, the CNN learns the concept of a *good* food arrangement based on people's choices. Our framework's input are the food ingredients for a given dish, and it starts with a random placement of them. Then, the CNN evaluates how good the initial arrangement is. If the evaluation is under a given threshold, then the system will move some of the ingredients according to our rearrangement algorithm. As it would be messy to be rearranging the actual food. The food arrangement is simulated using image processing. After a rearrangement has been done, the CNN will evaluate again how good the arrangement is. This will be repeated until the evaluation by the CNN surpasses a given threshold. Once the arrangement is good enough, the position of the food will be sent to the robot for its actual execution.

This paper is organized as follows: in section II we briefly review related work to cooking robots. In section III we describe our framework for arranging food on a plate. In section IV we show the results obtained using two different set of meals, and we show how a robot using a spatula-like gripper can place food on a plate as desired. Finally, in section V we give the conclusions of this paper.

II. Related Work

Regarding meal preparation, heretofore, robotics research have mainly focus on recognition and cutting of ingredients, tool manipulation, among others. Yamazaki et al. used a force

978-1-6654-1307-7/22/$31.00 ©2022 IEEE

sensor and simple image features to detect vegetables and a cutting board to cut them [4]. Kranz et al. implemented a networked sensing system to gather information about activities done in the kitchen, namely cutting different ingredients; their system is able to recognize what kind of ingredient was cut [5]. Kunze et al. developed a system that enables a robot to determine action parameters for cracking an egg, pouring liquids and making a pancake through the use of logic programming and simulation-based temporal projection [6], [7]. Haidu et al. developed a game for the extraction and learning of knowledge on how to pour and flip a pancake [8]. Bollini et al. proposed a system that is able to obtained recipes from the internet and translate them into robot motions for the robot to execute them [9]. Paul et al. [10] and Jelodar et al. [11] used CNNs for the classification and identification, respectively, of cooking object's states. The only similar work that we could find, uses Imitation Learning and an expert's food arrangement data for the robot to learn how to arrange the placement of Japanese Tempura [12]. However, their method requires the 3D model data of the ingredients to be arranged while our method uses simple RGB images and can handle several combinations of ingredients.

Regarding professional cooking arrangement, we consulted with a professional chef about the possible existence of rules. However, the answer was that there are no written rules but common sense in the color combination, overlapping, etc.

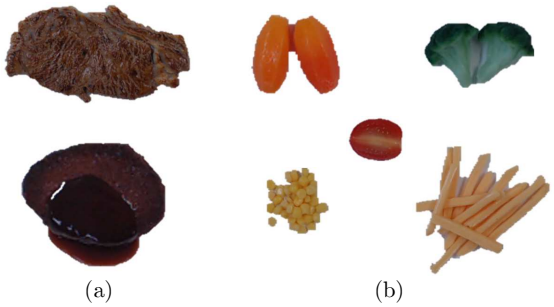
III. Food Arrangement Framework

In this section, we describe the proposed framework for serving a meal considering its presentation. Our framework is mainly composed of a CNN trained to evaluate how *good* a food arrangement is, and a rearrangement algorithm to modify the position of the food to increase the score of the arrangement.

A. Arrangement Evaluation

As explained in section I, we focus on how to transfer to the robot, the knowledge that humans have on how to arrange food on a plate. We start from the premise that making a *good* food presentation is part of common knowledge. We build the hypothesis that a CNN is able to learn a *good* food arrangement and therefore evaluate a given food arrangement by giving it a score. For the CNN to be able to give a numerical score, we need a dataset compose of images of food arrangements with their respective numerical score. We prepare a dataset with two main dishes: a steak and a Salisbury steak; and with three different garnishes to be picked from carrots, corn, french fries, tomatoes and broccoli, using food samples[1]. All the food samples have at least 3 different orientations/configurations, except for the steak, which due to its size, only one was used. Fig. 2 shows one example of each food used. The 2 main dishes with the 5 garnishes, gives us $2 \times \binom{5}{3} = 20$ different combinations of food arrangements. The data needed to train the CNN was collected using AMT. To avoid as much as possible a subjective evaluation, instead of asking people directly to give a score to each arrangement, we asked people to choose between a pair of arrangements with the same combination of food.

[1] https://en.wikipedia.org/wiki/Food_model



(a) (b)

Fig. 2: Examples of food sample images used to create the food arrangement dataset. In (a) main dish, and in (b) garnishes.

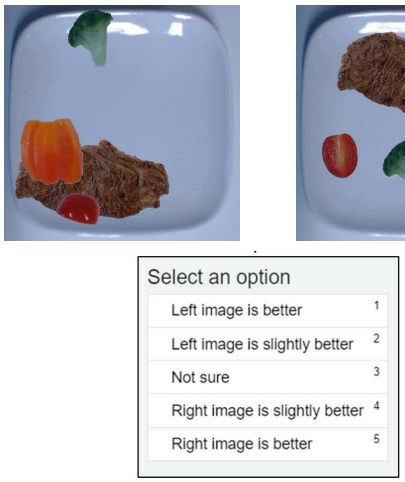


Fig. 3: An example of the questionnaire made in AMT.

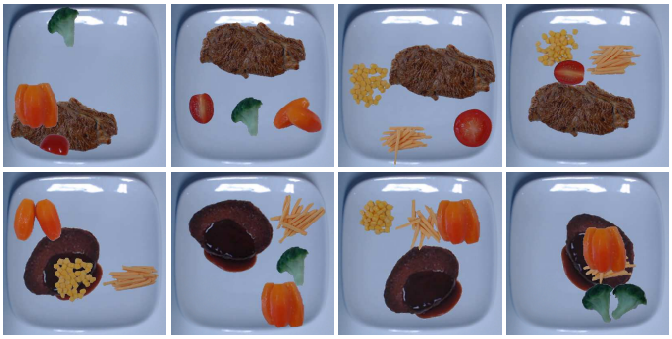


Fig. 4: Examples of the food arrangement images created. On the top row using the steak as main dish and on the bottom row using the Salisbury steak as main dish.

As shown in Fig. 3, we created a questionnaire with five choices, all of them relative to two images. We also asked the AMT's crowdworkers to avoid as much as possible using the option "not sure", since in previous recollection of data most of their answers were "not sure" (it is not easy to get people to answer seriously, since the faster they answer the faster they

TABLE I: Score given to each image according to the answer of the questionnaire.

Option	Left image	Right image
Left image is better	1	0
Left image is slightly better	0.75	0.25
Not sure	0.5	0.5
Right image is slightly better	0.25	0.75
Right image is better	0	1

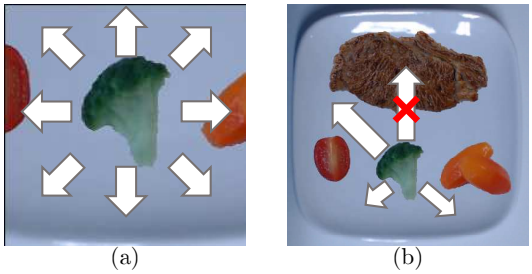
finish and get paid). The answers collected were converted to scores between 0 and 1 according to Table I. As we prepared 60 different arrangements per food combination, we obtained 59 relative evaluations for each arrangement. For each food combination, 25% of the arrangements were intentionally created as *good* arrangements. The rest were randomly created; i.e., the orientation/configuration of each garnish was randomly chosen and placed randomly. Fig. 4 shows some examples of the food arrangements used. We compute the average of these evaluations to obtained a single numerical score per food arrangement. Our dataset is composed of 1200 images. The 90% of the dataset is used for training, from which 10% is used for validation; the remaining 10% is used for testing.

We use a CNN with few layers for the sake of learning time. Our CNN is composed of 2 convolutional layers, followed by a max pooling layer, and another pair of convolutional-max pooling layers, that are connected to 2 fully connected layers, as illustrated in Fig. 1. The absolute error of the average over the test data obtained was about 0.1. The trained CNN will be used to evaluate food arrangements.

B. Rearrangement Algorithm

The algorithm implemented for rearranging food placed on a plate is shown in Algorithm 1. As inputs it receives the initial randomly generated food arrangement; i.e., an array of 2D positions of the top-left corner of the box bounding the food, and the food used as an array consisting of one main dish and 3 different garnishes. We set a maximum number of N loops for the algorithm to find a good arrangement. The algorithm first evaluates the initial arrangement and compares it to a given threshold, if the arrangement is below this threshold, we proceed to move one of the garnishes, since the main dish is never moved. With a 10% probability the algorithm will randomly chose one garnish and move it to three different places randomly over the space of the plate excluding the area of the main dish (`randomMove` function) without checking for overlapping with the other garnishes, as illustrated in Fig. 5(b). The `randomMove` function will return only the food position that yielded the best arrangement score out of the three places tested. Otherwise, with a 90% probability the algorithm will call the function `bestNeighbor`, for which we tested two different strategies.

In the first one, called `bestNeighbor-A`, the garnish to be moved is chosen randomly. Then, based on the garnish original position, we compute eight neighbors in eight different directions, as illustrated in Fig. 5(a), with a Euclidean distance of 40 pixels. From these positions we add a random position between -10 and 10 pixels in each axis for all the eight neighbors (directions). Then, we compute the score of the food arrangements using each of the eight neighbors positions for the garnish to be moved, except for those that lay outside the plate surface. The other two garnishes are left on their original positions. The function `bestNeighbor-A` will return the food position that yielded the best arrangement score.



(a) (b)

Fig. 5: Possible displacements of garnish. In (a) when using `bestNeighbor`, and in (b) when using `randomMove`.

In the second one, called `bestNeighbor-B`, the garnish to be moved is also chosen randomly. Then, based on the garnish original position, we compute eight neighbors using polar coordinates with θ's in the same eight different directions as in `bestNeighbor-A`, and r = `random`$(1, 50)$ pixels for each neighbor. Then, we compute the score of the food

Algorithm 1 Food rearrangement

$n \leftarrow 0$
$\mathbf{P}_0 \leftarrow$ initial 2D position of food
$\mathbf{F} \leftarrow$ type of food
$img \leftarrow$ CreateImage($\mathbf{P}_0$, $\mathbf{F}$)
$score \leftarrow$ Evaluate(img)
while $n < N$ **do**
  if $score \geq$ good **then break**
  else
    $action \leftarrow$ `random`$(0.0, 1.0)$
    if $action > 0.1$ **then**
      $\mathbf{P} \leftarrow$ `bestNeighbor`$(\mathbf{P}_0)$
    else
      $\mathbf{P} \leftarrow$ `randomMove`$(\mathbf{P}_0)$
      $action \leftarrow 0$
    end if
  end if
  $img \leftarrow$ CreateImage($\mathbf{P}$, $\mathbf{F}$)
  $newScore \leftarrow$ Evaluate(img)
  if $newScore > score$ **or** ($action = 0$ **and** `random`$(0.0, 1.0) < 0.5$ **and** $score < 0.7$) **then**
    $score \leftarrow newScore$
    $\mathbf{P}_0 \leftarrow \mathbf{P}$
  end if
  $n \leftarrow n + 1$
end while

arrangements using each of the eight neighbors positions, except for those that lay outside the plate surface. The function `bestNeighbor-B` will return the food position that yielded the best arrangement score.

After the call to `bestNeighbor` or `randomMove`, the algorithm will compute the $newScore$ of the food arrangement returned. If the $newScore$ is greater than the original $score$, the algorithm will update the positions of the food and the $score$ (lines: 20-III-B). Or, when the garnished was randomly moved and the $score$ is under 0.7, with a 50% of probability, the algorithm will update the positions of the food and the $score$. Otherwise, it will keep the original positions and $score$. Then, the loop begins again.

For the second loop onwards, we tested two different approaches to which we refer as "random" and "order". In the random approach, the garnished to be moved will always be chosen randomly. On the other hand, in the order approach, the `bestNeighbor` function will move the following garnish to that randomly chosen in the first loop. The order of the garnishes is given by the array $\mathbf{F}$ that contains the type of food. In subsequent loops ($n > 2$), it will move the following garnish to that of the previous loop. If it has already use all the three garnishes, it will go back to the first garnish moved and repeat until the obtained arrangement is updated (line III-B) or the algorithm finishes. However, at every loop after the food arrangement has been updated (i.e., there was an improvement in $score$ but it is still under the given threshold, line) the garnish to be moved will be picked randomly again. The aim of this approach is that if moving one garnish did not improved the arrangement score, then try with the next, instead of trying randomly. But, after having improve the arrangement, then try again randomly at first and from there go in order if needed.

It should be noted that the `randomMove` function will always randomly chose the garnish to be moved. The algorithm finishes when the food arrangement score is above or equal to the threshold given or when arriving to the maximum number of loops (N) allowed. The final food placement is sent to the robot, for the robot to place each food on its respective position on the plate, as illustrated in Fig. 1.

IV. Experimental Results

We carried out several tests to demonstrate the validity of the proposed food arrangement framework. We use 10 of the 20 possible combinations of main dish and garnishes, and ran 10 tests per combination. We also test the two strategies: `bestNeighborA` and `B`, and the two approaches: random and order, proposed for the rearrangement algorithm presented in section III-B, for each of the tests. For all the tests, we use $N = 100$ and the threshold for a *good* arrangement was set to 0.75. We recorded the initial score s_0 of the randomly generated arrangement and the final score s_f after the rearrangement algorithm finished. We also recorded the number of loops (ℓ_u) in which the food arrangement was updated—i.e., the number of loops that Algorithm 1 passes through line III-B—and the total number of loops ($\ell_t \leq N$) needed to generate a *good* arrangement in order to analyze the efficiency of the proposed algorithm.

TABLE II: Combinations of food used in the validation experiments.

Combination	Main dish	Garnishes		
		1	2	3
1	Steak	Carrot	Tomato	Broccoli
2	Steak	French fries	Carrot	Tomato
3	Steak	Corn	Carrot	Broccoli
4	Steak	Corn	Carrot	Tomato
5	Steak	Corn	French fries	Tomato
6	Salisbury	French fries	Carrot	Broccoli
7	Salisbury	Corn	Tomato	Broccoli
8	Salisbury	Corn	French fries	Broccoli
9	Salisbury	Corn	French fries	Tomato
10	Salisbury	Corn	French fries	Carrot

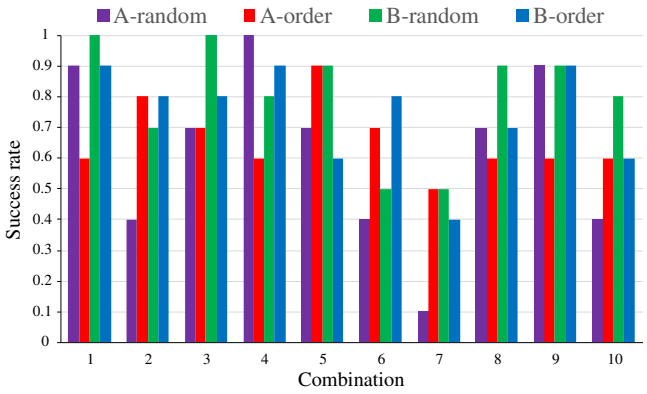


Fig. 6: Success rate average results over 10 tests per food combination per tested food arrangement strategy.

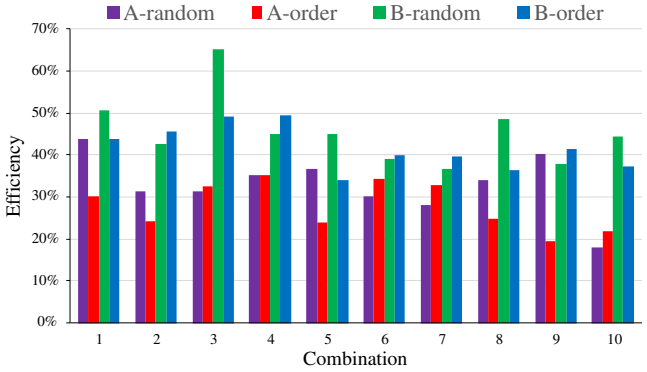


Fig. 7: Efficiency average results over 10 tests per food combination per tested food arrangement strategy.

The combinations used are summarized in Table II, where we chose 5 combinations per main dish, steak (S), and Salisbury steak (SS), and the garnishes corn (K), french fries (FF), carrot (C), tomato (T) and broccoli (B) as diverse as possible. The garnishes are in the order used in the array of type of food $\mathbf{F}$.

We tested the different strategies proposed in section III-B; i.e., when using function `bestNeighbor-A` and

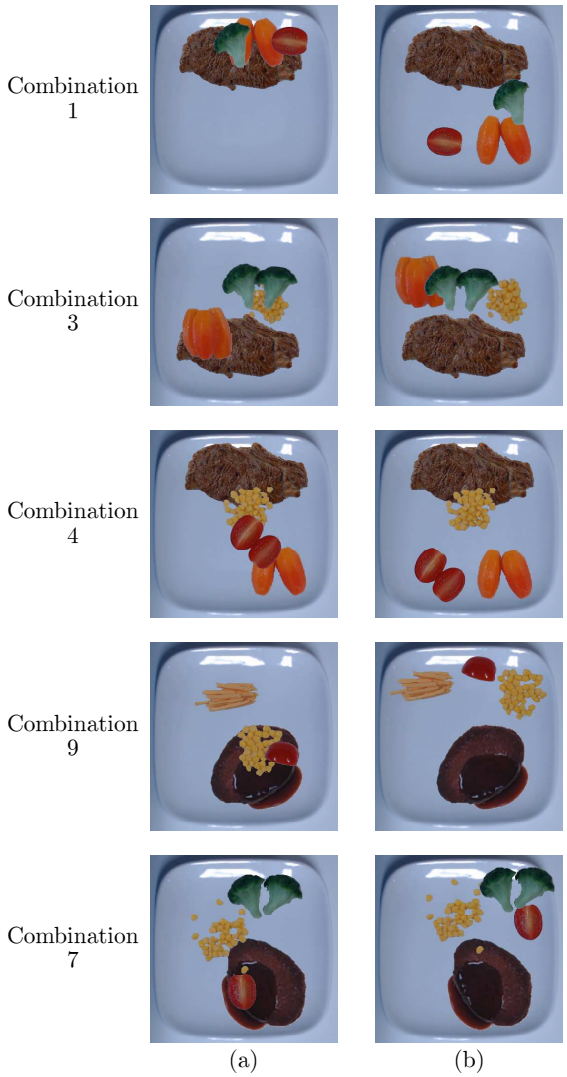


Fig. 8: Food arrangement results examples. In (a) the initial arrangement and in (b) the final arrangement.

the random approach (A-random), when using function `bestNeighbor-A` and the order approach (A-order), when using function `bestNeighbor-B` and the random approach (B-random) and when using function `bestNeighbor-B` and the order approach (B-order). The results obtained are summarized in the graphs shown in Figures 6 and 7 for the success rate and the efficiency ($\overline{\ell}_u/\overline{\ell}_t$ express in percentage), respectively. The numbers in the horizontal axis indicate the food combination used, as numbered in Table II. From these results, it can be seen that the function `bestNeighbor-B` in combination with the random approach yielded the best success rate overall tested food combinations, and also the best efficiency. It can also be observed that for some combinations the success rate was particularly low for all the tested methods, like No. 7 which was the worst, with an average over all methods of 0.37); while for combinations 1, 3, 4 and 9, the success rate was ≥ 0.8 on average over all the tested methods.

Fig. 8 shows the result of one test of each of the above mentioned food combinations when using the `bestNeighbor-B` in combination with the random approach. For combination 7, it can be seen that the corn configuration used, occupies a considerably large area of the plate, thus reducing the number of possible placements which makes it hard to rearrange in comparison with smaller garnishes like the broccoli or tomato, or with other corn configurations, like combinations 3 or 4. It can also be observed that overlap of less than a half of the garnish's surface among garnishes or between the garnish and the main dish was evaluated as *good* according to the collected dataset. Fig. 10 shows the food arrangements obtained at every updated loop using Combination 9, for the same test shown in Fig. 8. As it can be observed in (a), at the beginning the score is low since two of the three garnishes are overlapped with the main dish. The score improves as the corn is removed from the top of the main dish (b). In loop 4 (Fig. 10-e) a random action was taken and although the score is not greater than the previous one, the arrangement was updated (line: 19 of Algorithm 1). Then, as the tomato is removed from the top of the main dish (f), the score approaches the given threshold for a good arrangement. In the next loop (g), as the corn is moved horizontally towards the center of the plate, the score improves but not enough. Similarly, in the next loop (h), the corn is moved vertically towards the center of the plate which barely increases the score. Finally, in the last loop (i), the french fries are moved vertically towards the center of the plate which is good enough to increase the score above the given threshold. These means, that in the dataset collected, food arrangements with the garnishes near the edge of the plate were not evaluated as good as those with the garnishes placed towards the center of the plate.

Fig. 9 shows snapshots of a UR3 robot placing food on a plate according to the arrange generated with the proposed framework for Combination 1. The robot uses a spatula-like gripper [13] for picking the food samples. As it can be observed the robot is able to place the food on the desired locations; this means that we can use the proposed algorithm for preparing a *good* food arrangement using a simple gripper.

V. Conclusions

This paper proposed a food arrangement framework for a robot to carry out the final arrangement of food served on plates. We constructed a food arrangement dataset using food samples (i.e. fake food) and Amazon Mechanical Turk, to make a relative evaluation on how *good* food arrangements are, avoiding subjective evaluations. We used this dataset to train a CNN for numerically evaluating images of food arrangements. We proposed a rearrangement algorithm for generating a *good* food arrangement according to the CNN evaluation. Using ten different combinations of main dish and garnishes, we demonstrated the validity of the proposed algorithm, and showed that the function `bestNeighbor-B` in combination with the random approach yielded the best success rate among the tested methods. Finally, we showed that it is possible for

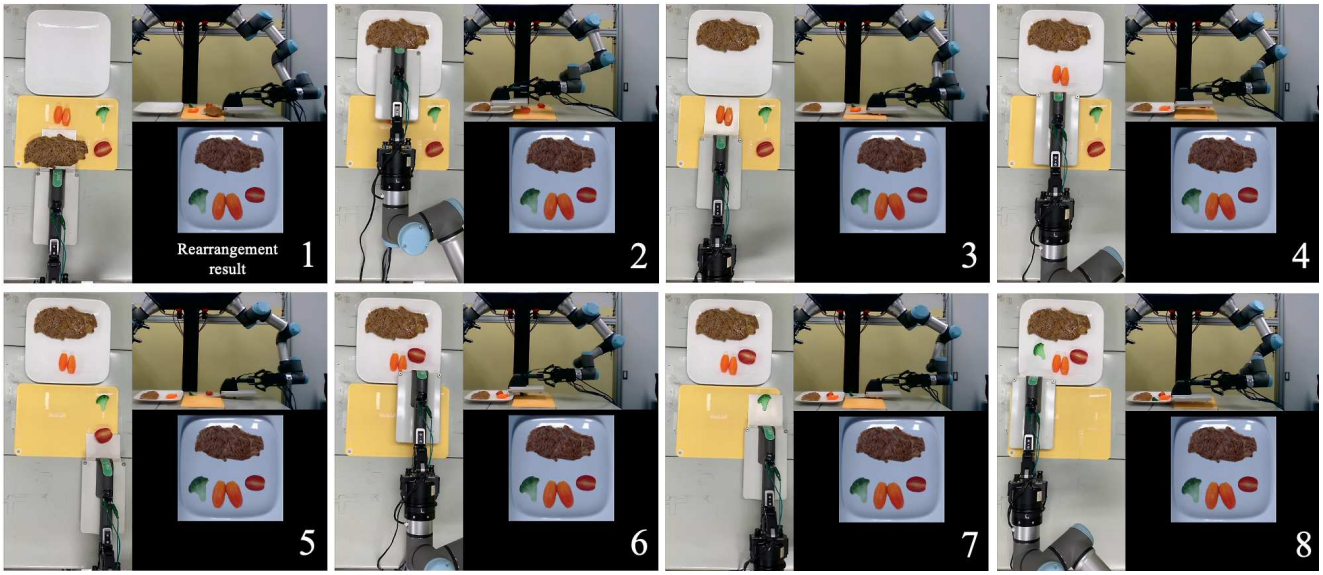
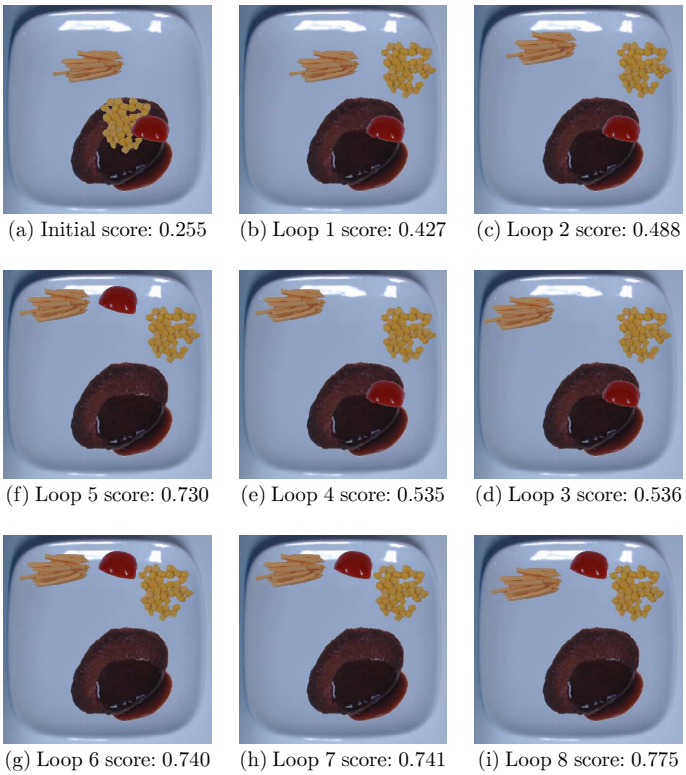


Fig. 9: Food placement by a UR3 robot when using food Combination 1.


(a) Initial score: 0.255 (b) Loop 1 score: 0.427 (c) Loop 2 score: 0.488

(f) Loop 5 score: 0.730 (e) Loop 4 score: 0.535 (d) Loop 3 score: 0.536

(g) Loop 6 score: 0.740 (h) Loop 7 score: 0.741 (i) Loop 8 score: 0.775

Fig. 10: Food rearrangement process when using food Combination 9.

a robot to place the food according to the food arrangement obtained with the proposed algorithm.

In the future, we would like to improved the accuracy of the CNN used to evaluate food arrangements, ideally gathering real pictures instead of using food samples, and increasing the variety of food used. We would also like to test our framework with unseen ingredients in the dataset. Finally, we would like to try reinforcement learning algorithms for the food rearrangement to be more efficient.

REFERENCES

[1] "Octochef (in japanese)," 2022, https://connected-robotics.com/products/octochef/.
[2] "Okonomiyaki robot," 2022, https://www.youtube.com/watch?v=nv7VUqPE8AE.
[3] C. Michel, C. Velasco, E. Gatti, and C. Spence, "A taste of kandinsky: assessing the influence of the artistic visual presentation of food on the dinning experience," *Flavour*, vol. 3, p. 7, 2014.
[4] K. Yamazaki, Y. Watanabe, K. Nagahama, K. Okada, and M. Inaba, "Recognition and manipulation integration for a daily assistive robot working on kitchen environments," in *Robotics and Biomimetics (ROBIO), 2010 IEEE Int. Conf. on*. IEEE, 2010, pp. 196–201.
[5] M. Kranz, A. Schmidt, R. Rusu, A. Maldonado, M. Beetz, B. Hornler, and G. Rigoll, "Sensing technologies and the player-middleware for context-awareness in kitchen environments," *Networked Sensing Sys 2007*, pp. 179–186, 2007.
[6] L. Kunze, M. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," in *The 10th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 1*. Int. Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 107–114.
[7] L. Kunze, M. Dolha, and M. Beetz, "Logic programming with simulation-based temporal projection for everyday robot object manipulation," in *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE, 2011, pp. 3172–3178.
[8] A. Haidu, D. Kohlsdorf, and M. Beetz, "Learning task outcome prediction for robot control from interactive environments," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6943183
[9] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus, "Interpreting and executing recipes with a cooking robot," in *Experimental Robotics*. Springer, 2013, pp. 481–495.
[10] R. Paul, "Classifying cooking object's state using a tuned vgg convolutional neural network," *arXiv preprint arXiv:1805.09391*, 2018.
[11] A. Jelodar, M. Salekin, and Y. Sun, "Identifying object states in cooking-related images," *arXiv preprint arXiv:1805.06956*, 2018.
[12] J. Matsuoka, Y. Tsurumine, Y. Kwon, T. Matsubara, T. Shimmura, and S. Kawamura, "Learning food-arrangement policies from raw images with generative adversarial imitation learning," in *2020 17th International Conference on Ubiquitous Robots (UR)*, 2020, pp. 93–98.
[13] "SWITL," 2022, https://www.furukawakikou.co.jp/switl/.