Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-18 (2025) 205–210

Data-driven Modeling and Optimization of Pinsa Dough Flattening with a Robot

Alessandro Colombo * **Riccardo Busetto** ** **Matteo Corno** *
Andrea Zanchettin * **Sergio Matteo Savaresi** *

* *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, 20133, Italy*
** *IDSIA Dalle Molle Institute for Artificial Intelligence USI-SUPSI, Lugano-Viganello, Switzerland.*

Abstract: Robotic automation in the food industry is being adopted more and more to satisfy the need for precision and efficiency. The ability to properly manipulate deformable objects plays a key role in food preparation, as ingredients frequently present plasto-elastic behaviors. This paper aims to develop a robotic system able to flatten the dough of a *Pinsa Romana*. The main elements of the developed system are a robotic arm and a depth camera: the 3D data from this camera is used to assess the state of the dough and determine the trajectories that the robot should follow to process the dough. By adopting a data-driven approach, we developed two algorithms that were able to achieve our objective by using a greedy strategy. These algorithms are then compared in terms of the number of actions needed to achieve the task and the total execution time.

Keywords: Deformable Object Manipulation, Food processing, Mechatronics, Neural networks, Robots manipulators.

1. INTRODUCTION

The increasing demand and the need for precision and reliability in the food industry has led to a steady growth in the adoption of robotic and autonomous solutions in this sector. However, most applications are still using rigid solutions, by employing machineries designed to fulfill a specific task. If a manufacturer were to differentiate their line of production, they would need to invest in new equipment. In this paper, we propose a more flexible solution adopting a collaborative robot, which could potentially be programmed to conduct multiple tasks.

Specifically, we developed a proof of concept for the flattening of a *Pinsa Romana* dough in its characteristic shape, aiming to match the work of a human operator. The main assets at our disposal are the aforementioned collaborative robotic arm and a depth camera. The robot manipulator presses on the dough mimicking human behavior and the camera is used to evaluate the flattening state. The proposed algorithms are used to appropriately decide the pressing locations.

Deformable object manipulation represents a critical area of research in the robotization of many sectors, due to the difficulty of keeping track of the plasto-elastic deformation of soft matter. Objects in this category can be classified by their shape (Sanchez et al. (2018)). For instance, we can have linear objects like cables and ropes (e.g. Cao et al. (2024), Liu et al. (2023), Monguzzi et al. (2025)), planar objects like cloths and sheets (e.g. Zhao et al. (2024), Zhang et al. (2024), Shehawy et al. (2023)), and volumetric objects like plush toys, soft organic tissue (e.g. Shen et al. (2022), Liang et al. (2024)) and food products. The focus of this research is on the last.

Similarly to our task, Figueroa et al. (2016) use a rolling pin end effector to spread a pizza dough. Optimal robot trajectories are learned from human demonstration. In Petit et al. (2017), focus is put on the deformation tracking of a "pizza-like" material with known mechanical properties using 3D vision, by modeling it via FEM; this approach provides high fidelity, but is computationally demanding.

In more recent works, a model free approach is preferred, with the object behavior being learned from data. Bartsch et al. (2024) and Li et al. (2024) both developed a robotic station that molds small blocks of clay into desired shapes using simple electric grippers. Multiple depth cameras are used to reconstruct the surface of the block to track deformations. A 1-step predictor (a pre-trained pointcloud reconstruction transformer and a recurrent state-space model respectively) provides an estimate of the effect of a given action, allowing for an MPC approach.

Shi et al. (2023) prepares dumplings starting from a piece of dough by decomposing the procedure into small simpler tasks (e.g. cutting, rolling, etc.), each using its own dedicated end effector, starting from a human demonstration. The presented framework is then validated by using those tools to shape cookie dough into letters of the alphabet.

A common trait of these works is a specific 3D representation of the desired shape, so that a cost function can clearly quantify the distance between the real object shape and the desired one. By design, Pinsa does not have a regular desired shape: its only requisites are that it is oblong, that its surface is irregular, with recesses and bubbles, and that

10.1016/j.ifacol.2025.10.221

it is below a certain thickness. Human workers achieve this by pressing vertically on the dough with spread fingers, back and forth along the spine of the dough.

Our focus for the control algorithms in this work will be on the last point, guaranteeing the dough to be below a threshold thickness, while the other two will be obtained through the set-up design: the oblong shape will be obtained by using an oval starting dough and operating along the dough principal axis, while the rugged surface is achieved using a hand-shaped end-effector.

The main contribution of this paper are: (i) we present two algorithms that allow our robotic system to spread the dough below a given thickness: the first adopts a greedy strategy, based on the result of an analysis of the dough behavior, the second builds upon the first by introducing a 1-step predictor to reduce execution time; (ii) we experimentally validate the performance of these algorithms and compare them to a benchmark

The rest of the paper is structured as follows. In Section 2, we formally present the problem statement. In Section 3 we describe the overall methodology applied to achieve our task. In Section 4, we show an analysis of the effect of the pressing action and outline the reasoning behind our greedy strategy. Section 5 describes the 1-step predictor we developed for our second algorithm. Section 6 contains an overview of the experimental set-up and its components. Section 7 shows the experimental results for our algorithms and compares them with benchmark ones. Finally, Section 8 summarizes the conclusions of our research.

2. PROBLEM STATEMENT

The objective of this paper is to develop a robotic station able to flatten a dough of Pinsa. The dough is considered flattened if its thickness, as in its height H is below a threshold $H_{th} = 2.5cm$. Furthermore, we want to minimize first the number of pressing actions required to complete the flattening, and second the overall execution time. The former objective in particular is related to the organoleptic properties of pinsa dough, as an excessive amount of interaction with it can damage its gluten network.

As such, defining s_0 as the initial state of the dough, a_k the k^{th} action, N_a the total number of pressing actions used to reduce the dough height below H_{th}, $s_{k+1} = T(s_k, a_k)$ for $k = 0, \dots, N_a - 1$ the state transition law, and $H_k = \mathcal{F}(s_k)$ a function that computes the dough height, our problems can be written as:

$$\begin{aligned} &\min_{a_0,\dots,a_{N_a}-1} N_a \\ &\text{subject to:} \\ &\quad s_{k+1} = T(s_k, a_k) \text{ for } k = 0, \dots, N_a - 1 \\ &\quad \mathcal{F}(s_{N_a}) \leq H_{th} \end{aligned} \tag{1}$$

3. METHODOLOGY

The key elements of the system are: an anthropomorphic robot manipulator fit with an end-effector able to replicate the human pressing action, a depth camera placed in a fixed position that monitors the scene, and a piece of dough, initially placed in a random position on a horizontal work plane.

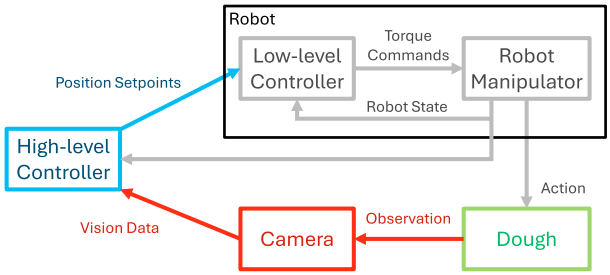


Fig. 1. Control scheme

Algorithm 1 General algorithm

Capture pointcloud Pcd_k
Evaluate pointcloud height H_k
while $H_k > H_{th}$ **do**
  Compute the pressing point list $\mathbf{P_{list}}$
  for $P \in \mathbf{P_{list}}$ **do**
    Press on point P
  Move to *Home* position
  Capture pointcloud Pcd_k
  Evaluate pointcloud height H_k

The overall control architecture is of the Position-Based Dynamic Look-and-Move type: 3D visual data from the camera, in the form of a pointcloud, is used to compute the reference trajectory for the robot to follow; the robotic arm is considered as an ideal positioner, meaning that we rely on its precision and repeatability. As a result, the control scheme consists of two nested loops: the inner one, managed by the robot servomotors, tracks a sequence of reference poses, and the outer one, managed by the high-level controller, computes those trajectories based on the visual sensors data. (see Fig. 1).

The overall procedure takes the following form (See Algorithm 1):

(a) A pointcloud pcd_k of the dough is captured, and the state of the dough is inferred.
(b) The height of the dough H_k is evaluated as detailed in the following.
(c) The high-level controller select the appropriate pressing point(s) (see Section 4) and computes the target positions.
(d) The robot moves to a *Home* position so that new vision data can be captured, moving back to step (1).

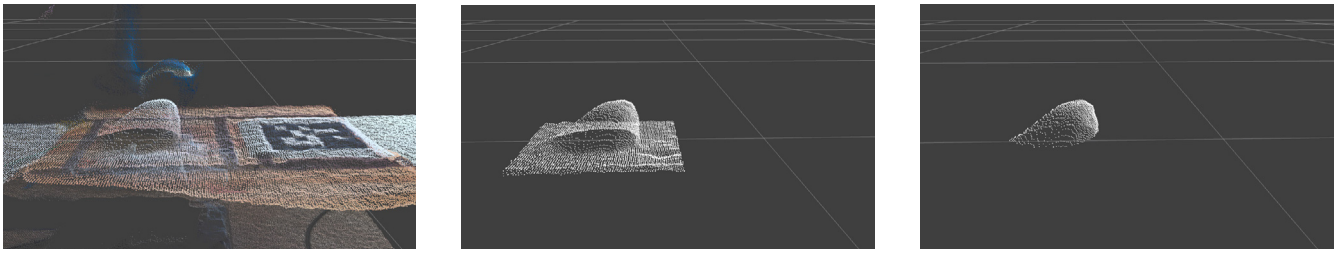
3.1 Pointcloud segmentation

The segmentation process purpose is to extract information about the dough state from the camera pointcloud.

To isolate the points of belonging to the dough, we follow these steps (see Fig.2):

(a) We transform the coordinates of the pointcloud points into a reference frame fixed with the location where the dough is placed, defined *board* frame.
(b) We isolate a Region Of Interest (ROI) by setting limits on the x,y, and z values of the points.
(c) We remove the points belonging to the ground by applying the RANSAC algorithm (Derpanis (2010))

The remaining points belong to the surface of the dough as seen from the camera.



(a) Starting pointcloud (b) Region of Interest (c) Segmented pointcloud

Fig. 2. Segmentation steps

We define the height of the dough as

$$H = z_{max} = \max_i z_i$$

where z_i is the height of generic point $p_i = (x_i,\ y_i,\ z_i)$ belonging to the segmented pointcloud.

Furthermore, we can compute the principal axis of the dough by performing Principal Component Analysis (Pearson (1901)) on the projection of the dough points on the xy plane. The main direction is used as reference for the alignment of the robot motion.

3.2 Robot Motion

Robot motion in our algorithms consists in two main actions: pressing on a given point (*Press* in Algorithm 1), and clearing the working area to allow visual inspection (moving to the *Home* position). The former is the building block of our algorithms: it consists in a motion primitive that allows the robot to press on a specific given point. It is composed of three steps: reach, press, and clear.

Considering the tip of the robot end effector as tool center point (TCP), the first motion brings it directly above the requested pressing point. The second moves it down until the TCP touches the ground at the coordinates of the requested pressing point. Finally, the third brings the end-effector back to the position at the end of the first step.

The end-effector keeps the same orientation throughout this movement, which is fixed with respect to the dough spine, computed in the previous section. The entire motion primitive is defined by the xy coordinates of the pressing point. From that, the other position references are computed accordingly.

The *Home* position is selected so that the depth camera can have a clear view of the area where the dough is placed, without obstruction from the end-effector. Once the robot reaches the *home* position, a command is sent to the high-level controller signaling that the visual data does not contain any occlusion.

4. ANALYSIS OF THE PRESSING ACTION EFFECT

The core of the two algorithms developed in this work consists in selecting where to press. In order to better understand the mechanics involved, an analysis of the effects of the pressing action was performed.

A dataset was then collected, each sample composed of the segmented pointcloud before and after the pressing action, and the coordinates of the pressing point.

In order to make the two pointclouds comparable, both are interpolated over the same grid on the xy plane using Delaunay triangulation, so that we can evaluate their difference by looking at their z coordinates. The grid is centered in the pressing point, and covers a 32.5cm sided square, with a 1mm step. The obtained points are then binned into 5x5mm squares, forming a set of 65x65 squares. we can then define a matrix Z so that its element (i, j) represents the height of the square in the j^{th} row and k^{th} column. The matrix associated with the pointcloud before the pressing action is defined as Z_b, the one after Z_a. Matrix $Z_\Delta = Z_a - Z_b$ represents the difference in height of the two pointclouds. Furthermore, based on the camera noise, we select 1.5mm as the minimum visible effect, and isolate the local effect of the pressing action, by defining

$$Z'_\Delta(j,k) = \begin{cases} Z_a(j,k) - Z_b(j,k), & if\ C(i,j) \\ 0 & otherwise \end{cases}$$

where

$$\begin{aligned} C(i,j) = \quad & Z_\Delta(j,k) \leq -1.5mm \wedge & \text{(a)} \\ & \wedge\, Z_a(j,k) \geq 0.5mm \wedge & \text{(b)} \\ & \wedge\, Z_b(j,k) \geq 0.5mm & \text{(c)} \end{aligned}$$

Condition (a) filters out negligible differences in height. Conditions (b) and (c) removes noise due to horizontal displacement of mass: when pressing vertically on the dough, it tends to expand or slightly move across the board; if $Z_b(j,k) \leq 0.5mm$ or $Z_a(j,k) \leq 0.5mm$, it means that there was no material at those coordinates, either before or after the pressing action respectively. Therefore, any difference value seen in those squares is due to horizontal motion only and thus ignored.

We then consider only the N_Z nonzero elements of $Z'_\Delta(j,k)$, $\Delta z_1, \dots \Delta z_{N_z}$, which allow us to evaluate the area affected by the press

$$A_{press} = A_{square} \times N_Z, \tag{2}$$

where $A_{square} = 25mm^2$ is the area of a single square, and the average absolute height difference in the affected area

$$|\Delta z_{avg}| = \left| \frac{1}{N} \sum_{j=1}^{N} \Delta z_j \right| \tag{3}$$

These quantities are compared with z_{init}, the initial height of the square corresponding to the pressing point.

Results for this analysis can be seen in Fig. 3: both quantities tend to grow with respect to z_{init}.

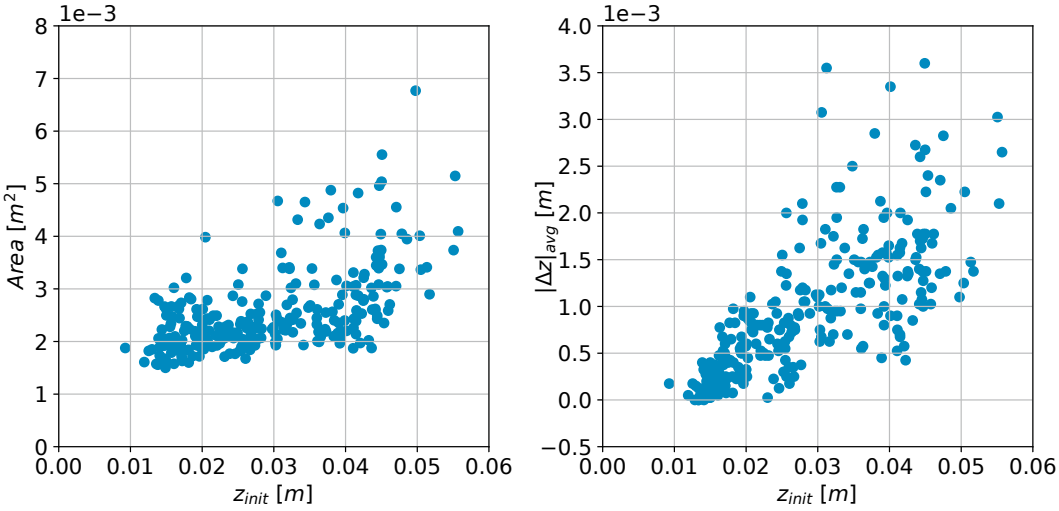


Fig. 3. Affected area and average height decrease with respect to the initial height of the pressing point

Algorithm 2 Greedy algorithm

Capture pointcloud Pcd_k
Evaluate pointcloud height H_k
while $H_k > H_{th}$ **do**
  Compute the highest point $P_{z_{max}}$
  Press on point $P_{z_{max}}$
  Move to *Home* position
  Capture pointcloud Pcd_k
  Evaluate pointcloud height H_k

Given these considerations, it seems that the higher the z coordinate of the pressing point, the wider the area of effect (r=0.73) and the deeper the height decrease (r=0.58). As such, our strategy is a greedy one: at each iteration, we press on the highest point of the dough to obtain the maximum effect (see Algorithm 2).

One limitation of this approach is the fact that we need to capture a new pointcloud after each pressing action to find the next highest point. This translates into longer execution times, as pointcloud processing can be slow. To increase the process speed, while maintaining the same strategy, we must be able to predict the location of the next highest point.

5. HIGHEST POINT PREDICTION

In order to reduce the overall execution time of the algorithm presented in the previous section, a Neural Network(NN)-based one-step predictor was developed to estimate the location of the next highest point.

The Feed-forward Neural Network used for this prediction receives as input a representation of the dough pointcloud and the coordinates of the pressing point and outputs the coordinates of the next pressing point.

To train this NN, the same dataset seen in the previous section was used, split into training and testing dataset, but with a slightly different approach. In fact, we interpolate the pointcloud over a 35x35cm square grid with a 1mm step, centered on the pointcloud center

Algorithm 3 NN-Greedy algorithm

Capture pointcloud Pcd_k
Evaluate pointcloud height H_k
while $H_k > H_{th}$ **do**
  Compute the highest point $P_{z_{max}}$
  Predict the next highest point $\hat{P}_{z_{max}}$
  Press on point $P_{z_{max}}$
  Press on point $\hat{P}_{z_{max}}$
  Move to *Home* position
  Capture pointcloud Pcd_k
  Evaluate pointcloud height H_k

$$(x_c,\ y_c) = \left(\frac{\sum_{i=1}^{N_p} x_i}{N_p},\ \frac{\sum_{i=1}^{N_p} y_i}{N_p} \right)$$

where $(x_i,\ y_i)$ are the xy coordinates of generic point i, and N_p is the total number of points in the pointcloud.

The resulting points are then binned into 1cm squares, from which we obtain the 35x35 height matrix Z_b.

The inputs of the NN consist in all the elements of Z_b, plus the xy coordinates of the pressing point $(x_p,\ y_p)$, for a total of 1227 elements; the output are the xy coordinates of the highest point of the pointcloud after the pressing action.

Hence, the NN is composed of an input layer with 1227 nodes, a hidden layer with 128 nodes, and an output layer with 2 nodes. The size of the hidden layer was chosen empirically by observing the behavior of the loss function during training, for both the training and testing dataset, by finding a compromise between low overfitting and low loss for the training data.

ReLu was used as activation function for the nodes. For training, MSE was chosen as loss function, and Adam as the optimizer.

After training, the NN could predict the next highest point with an average error of 2.3cm. In Algorithm 3, we can see how the NN allows us to press twice per pointcloud capture.

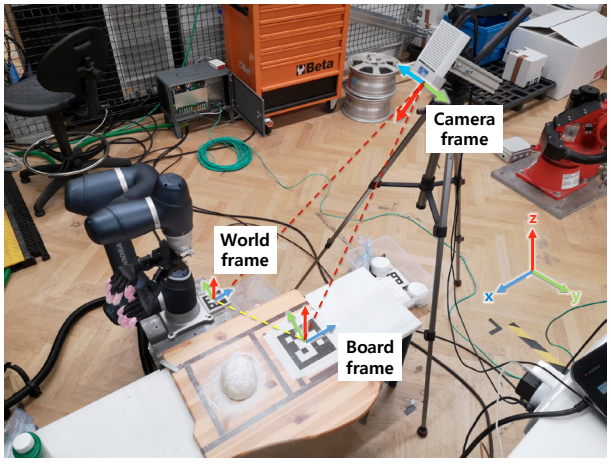


Fig. 4. Set-up for experimental validation (robot, a board to be considered as working surface, and a depth camera). The relevant reference frames are labeled.

Multi-step predictors to further estimate successive highest points unfortunately proved to be unreliable, and were discarded.

6. EXPERIMENTAL SET-UP

In our experiments, we used a Doosan Robotics cobot model A0509s, an Azure Kinect depth camera, a chunk of soft plastic material, and two fiducial markers (see Fig.4).

The two ArUco fiducial markers are placed in known and appropriate positions: one over the board on which the dough is placed, and one beside the base of the robot. From the camera 2D image it is possible to compute the relative pose of the camera, the board, and the robot. Therefore, we can transform coordinates in space between the *world* (the robot) reference frame, the *board* reference frame, and the *camera* reference frame. In fact, the robot's position setpoints are passed to the low level controller in the first frame, the points selected by the high level controller are defined in the second, and the coordinates of the pointcloud points are set in the third.

The robot wrist is equipped with a passive end-effector (see Fig. 5). It aims at replicating the hands position of a human worker. It is composed of three pieces: a base to connect it to the robot's wrist, and two hands, which are able to rotate 30° from their rest position but are kept there in place by two return springs. The purpose of the spring is to reduce the mechanical stress applied over the fingers when pressing on the dough. In the interest of keeping the focus of this paper on the strategical selection of the pressing points, the pressing action is performed using a single hand (the right one). The tool center point for the pressing action is the tip of the hand's middle finger, which is the first contact point between the tool and the dough.

7. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithms, we measured the total execution time and required number of press actions needed to flatten the dough, down to 2.5cm height.

Fig. 5. Custom made end-effector

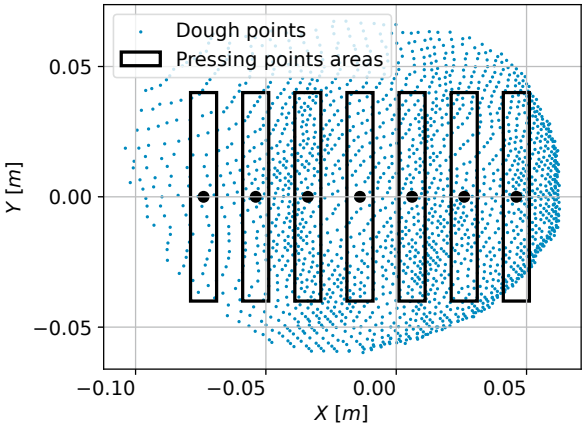


Fig. 6. Possible pressing areas for the human-like algorithms

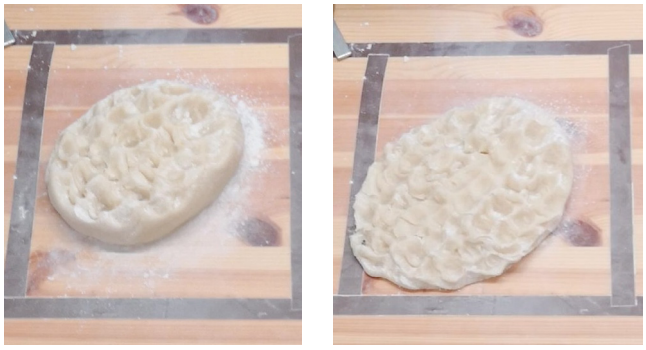
As benchmark, two more algorithms were deployed: the first (Human-like) mimics the motion of a human operator, and is used to validate the efficiency of our greedy algorithm; the second one (Greedy-random) presses on the highest point of the dough and then on a random point of the dough, and is used to test the effectiveness of our predictor.

Specifically, the human-like algorithm consists in selecting multiple pressing points from each pointcloud scans: each pressing point is selected along the spine of the dough, at a distance of 2cm from each other, then a random xy offset is added to avoid pressing over the same points. In Fig. 6, the possible pressing areas for a given pointcloud instance (projected on the xy plane) are shown.

The greed-random approach is functionally equivalent to Algorithm 3, but the second pressing point is chosen randomly across the dough surface.

Table 1 shows the results of our experiments in terms of execution time, number of individual pressing actions, and frequency of presses.

We can see how the greedy algorithm requires fewer pressing actions than the human-like one, meaning that each of the former's pressing actions is more efficient in flattening the dough. This can be attributed to the fact that the pressing points in the latter are selected irrespective of the height of the dough in that area, thus potentially pressing where where the dough was already



(a) (b)

Fig. 7. Final result for the Greedy algorithm (a) and for the Human-like one (b)

Table 1. Experimental results

Algorithm	N_a	T [s]	N_a/s [Hz]
Greedy	**30**	237	0.1266
NN-Greedy	32	183	0.1749
Human-like	40	**140**	**0.2857**
Greedy-random	40	228	0.1389

below the 2.5cm threshold. In Fig. 7 the resulting spread dough from the two algorithms are shown. In the one produced by the human-like algorithm, the dough is spread more than necessary.

The greedy algorithm is however also the slowest, due to its low frequency of press actions. The NN-based greedy algorithm improves the execution time, while maintaining a number of pressing actions similar to the one of the original greedy algorithm. Seeing that the greedy-random approach requires more pressing actions, we can further infer that our NN can accurately predict the next highest point.

8. CONCLUSIONS

In this paper, we developed 2 algorithms in order to flatten a dough of pinsa: the first follows a greedy strategy, by pressing on the highest point seen by a depth camera; the second builds on the first by predicting the next highest point using a feed-forward Neural Network, reducing the execution time by pressing twice per pointcloud capture.

Both algorithms were compared to benchmark algorithms in order to assess their quality. Results showed how our algorithms were able to spread the dough with a lower number of pressing action with respect to the benchmark ones.

REFERENCES

Bartsch, A., Avra, C., and Farimani, A.B. (2024). Sculptbot: Pre-trained models for 3d deformable object manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 12548–12555. IEEE.

Cao, B., Zang, X., Zhang, X., Chen, Z., Li, S., and Zhao, J. (2024). Shape control of elastic deformable linear objects for robotic cable assembly. *Advanced Intelligent Systems*, 2300835.

Derpanis, K.G. (2010). Overview of the ransac algorithm. *Image Rochester NY*, 4(1), 2–3.

Figueroa et al. (2016). Learning complex sequential tasks from demonstration: A pizza dough rolling case study. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 611–612.

Li, C., Ai, Z., Wu, T., Li, X., Ding, W., and Xu, H. (2024). Deformnet: Latent space modeling and dynamics prediction for deformable object manipulation. *arXiv preprint arXiv:2402.07648*.

Liang, X., Liu, F., Zhang, Y., Li, Y., Lin, S., and Yip, M. (2024). Real-to-sim deformable object manipulation: Optimizing physics models with residual mappings for robotic surgery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 15471–15477. IEEE.

Liu, F., Su, E., Lu, J., Li, M., and Yip, M.C. (2023). Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. *IEEE Robotics and Automation Letters*, 8(7), 3964–3971.

Monguzzi, A., Dotti, T., Fattorelli, L., Zanchettin, A.M., and Rocco, P. (2025). Optimal model-based path planning for the robotic manipulation of deformable linear objects. *Robotics and Computer-Integrated Manufacturing*, 92, 102891.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), 559–572.

Petit et al. (2017). Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot. *Robotics and Autonomous Systems*, 88, 187–201.

Sanchez, J., Corrales, J.A., Bouzgarrou, B.C., and Mezouar, Y. (2018). Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37(7), 688–716.

Shehawy, H., Pareyson, D., Caruso, V., De Bernardi, S., Zanchettin, A.M., and Rocco, P. (2023). Flattening and folding towels with a single-arm robot based on reinforcement learning. *Robotics and Autonomous Systems*, 169, 104506.

Shen, B., Jiang, Z., Choy, C., Savarese, S., Guibas, L.J., Anandkumar, A., and Zhu, Y. (2022). Acid: Action-conditional implicit visual dynamics for deformable object manipulation. *Robotics: Science and Systems XVIII*.

Shi, H., Xu, H., Clarke, S., Li, Y., and Wu, J. (2023). Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. In *Conference on Robot Learning*, 642–660. PMLR.

Zhang, Y., Liu, F., Liang, X., and Yip, M. (2024). Achieving autonomous cloth manipulation with optimal control via differentiable physics-aware regularization and safety constraints. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 9931–9938. IEEE.

Zhao, C., Jiang, C., Luo, L., Yuan, S., Chen, Q., and Yu, H. (2024). Learning thin deformable object manipulation with a multi-sensory integrated soft hand. *arXiv e-prints*, arXiv–2411.