Downloaded from orbit.dtu.dk on: Feb 21, 2025

DTU Library

An Adaptive Robotic System for Doing Pick and Place Operations with Deformable Objects

Jørgensen, Troels Bo; Jensen, Sebastian Hoppe Nesgaard; Aanæs, Henrik; Hansen, Niels Worsøe; Krüger, Norbert

Published in:
Journal of Intelligent and Robotic Systems: Theory and Applications

Link to article, DOI:
10.1007/s10846-018-0958-6

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Jørgensen, T. B., Jensen, S. H. N., Aanæs, H., Hansen, N. W., & Krüger, N. (2019). An Adaptive Robotic System for Doing Pick and Place Operations with Deformable Objects. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *94*(1), 81-100. https://doi.org/10.1007/s10846-018-0958-6

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Journal of Intelligent & Robotic Systems (2019) 94:81–100
https://doi.org/10.1007/s10846-018-0958-6

An Adaptive Robotic System for Doing Pick and Place Operations with Deformable Objects

Troels Bo Jørgensen[1] · Sebastian Hoppe Nesgaard Jensen[2] · Henrik Aanæs[2] · Niels Worsøe Hansen[3] · Norbert Krüger[1]

Received: 3 January 2018 / Accepted: 12 November 2018 / Published online: 3 December 2018
© Springer Nature B.V. 2018

Abstract

This paper presents a robot system for performing pick and place operations with deformable objects. The system uses a structured light scanner to capture a point cloud of the object to be grasped. This point cloud is then analyzed to determine a pick and place action. Finally, the determined action is executed by the robot to solve the task. The robotic placement strategy contains several free parameters, which should be chosen in a context-specific manner. To determine these parameters we rely on simulation-based optimization of the individual use cases. The entire system is tested extensively in real world trials. First, the reliability of the grasp is evaluated for 7 different types of pork cuts. Then the validity of the simulation-based optimization of the placement strategy is evaluated for 2 of the most different pork cuts, to show the generality of the overall approach.

Keywords Robotic manipulation · Deformable objects · Structured light scanner · Vision-based meat analysis · Simulation-based optimization

1 Introduction

Minimizing setup times for industrial robotic systems is an important task for incorporating robots in small batch production, since designing and integrating the system is a relatively large part of the total expense in this type of production. In this paper, we focus on meat handling, where we investigate the possibilities for using robots to execute pick and place operations of meat pieces. The challenge is that there are a lot of different cuts of meat, and special solutions have to be designed for each case. Thus it is important that a procedure is formulated, which can help to design robotic solutions for as many cases as possible in a reasonable amount of time.

We approach this problem from two directions. First, we design a general and adaptable hardware setup for doing pick and place operations with meat. Secondly, we present a simulation-based optimization framework for designing and fine tuning the solution in simulation.

The hardware setup is shown in Fig. 1 and it consists of a 6 axis robot arm, a suction-based gripper tool and a vision system. The gripper tool can be adapted to a specific task and it is designed to cope with the high variation when grasping deformable objects. The vision system is also designed to be a generic solution for detecting and segmenting meat surfaces. It uses stereoscopic structured light for 3D surface reconstruction, which has been shown to generate precise point clouds, even when scanning materials with high levels of subsurface scattering [16]. Furthermore, we apply a generic region growing method to segment the individual meat surfaces, which we have applied successfully to different cuts of pork. The

✉ Troels Bo Jørgensen
trjoe@mmmi.sdu.dk

Sebastian Hoppe Nesgaard Jensen
snje@dtu.dk

Henrik Aanæs
aanes@dtu.dk

Niels Worsøe Hansen
nwh@dti.dk

[1] Maersk McKinney Møller Institute, University of Southern Denmark, 5230 Odense M, Denmark

[2] DTU Compute, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

[3] Danish Meat Research Institute, Danish Technological Institute, 2630 Taastrup, Denmark

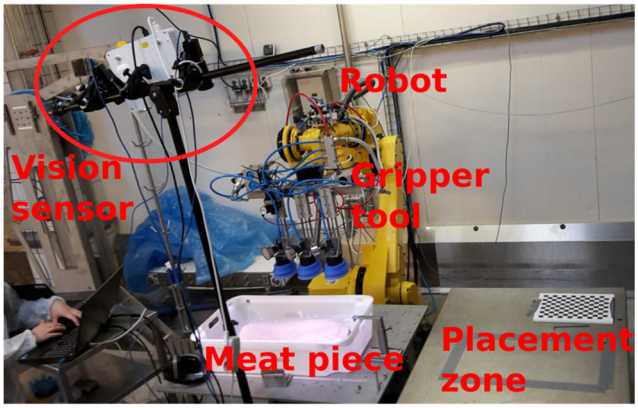


Fig. 1 The physical prototype used to handle meat pieces

segmented point cloud is then used to generate a robotic action for lifting and placing the meat. This action is also designed such that it can be adapted to specific use cases.

To adapt the generic solutions to specific use cases, we have designed a simulation tool for modeling the robotic meat handling operations. Furthermore, the simulation framework enables the user to analyze the robustness of the solutions. This is achieved by doing hundreds of simulations with different perturbations of system uncertainties, e.g. the meat size, to ensure the system works even for products with high variation, such as meat. We parameterized the generic solutions, such that they can be tuned for the specific problems using numeric optimization. The optimization is done based on the simulation framework, such that ten thousands of simulations are used to determine good system parameters. After good parameters are found in simulation, they are implemented and evaluated in the real world.

The main contribution of this work lies in combining several technologies, in order to design a robot solution for handling pork in a physical prototype at a Danish slaughterhouse. The individual technologies have been published in various conference proceedings. The gripper and the grasping strategy used in this work was introduced in [19]. This is extended by parameterizing the grasp action and introducing a placement strategy. The simulation framework used was introduced in [17]. In our work, this framework is extended to model the use cases addressed in this paper. Lastly, the optimization approach is based on work presented in [18].

The paper is structured as follows: First we discuss relevant litterature addressing the three main components in Section 2. These components are vision solutions for segmenting meat, robot systems for handling meat and optimization techniques relevant for robotic systems. The overall system is described in Section 3. The vision system for generating and segmenting the point clouds is described in Section 4. In Section 5, we introduce the gripper tool and discuss the procedure for generating a robotic action based on the point cloud. The case specific tuning of parameters is split into two parts. First, we introduce the simulation tool in Section 6 and then the optimization process is described in Section 7. In Section 8, we test the solutions with different cuts of pork. Lastly, we conclude on the results in Section 9.

2 Related Work

In this paper, 3 key topics are addressed. The first topic is vision based segmentation of meat surfaces. The second topic is robotic systems for manipulating deformable objects such as meat. Lastly, optimization based parameter tuning of robotic systems is addressed.

2.1 Vision Systems for Analysing Deformable Objects

3D reconstruction and simulation of deformable objects and has been studied intensively for years. A recent example would be [23] where cloth is handled dynamically by a humanoid robot. Here a control algorithm is fed input data from a Kinect that supplies both color and depth. Similar approaches can be found in [3, 34] and [24]. Common for these is that the object of interest is distinct and easily segmented from its environment. As such they are not directly applicable to our problem domain. This is because we have to handle boxes of meat with multiple pieces of meat in a pile. For this reason, depth data supplied is too inaccurate to properly segment each piece.

One needs to look no further than the DAVIS challenge [30] to see the tremendous progress and challenge of object segmentation. Some researchers have proposed to used convolutional neural network [5, 37], others pursue other strategies such as region augmentation via Gaussian mixture models (GMM) [22]. However, while they focus and succeed at segmenting a single primary object, they do not consider a cluttered scenario as our system will have to deal with.

Our contribution will be applying high accuracy depth from structured light and a simple, yet powerful segmentation algorithm to obtain depth data for each piece of meat. The superior accuracy [9] of stereoscopic vision enables us to distinguish individual pieces, something that would likely be impossible with the Kinect.

2.2 Robotic Solutions for Handling Meat

While a huge body of work has addressed pick and place operations for rigid objects, only limited research has addressed these operations for deformable objects, such as meat. One example is [3] who developed a robotic system for handling silicon elements which was used as a more test friendly replacement for meat. They both addressed peg-in-hole operations and laying down operations of deformable

objects with their system. In this work, we focus on real world cases and use substantially different equipment to address the grasping challenges of real meat products.

For related tasks such as cutting and separation of meat pieces research has been done in [25, 29] and [28]. Long et al. [25] proposed a system using three robots, one for moving the vision system, one for holding the meat and one for cutting the meat. Furthermore, they developed a simulator for modeling the deformable meat handling operation. Nabil et al. [29] proposed a similar system, but focused more on physically accurate simulation of the use case. Our proposed approach similarly rely on simulation-based analysis of the problem. However, we focus on modeling the interactions between the meat and its surroundings rather than just the interaction with a knife. We also use numeric optimization to tune the solutions in simulation, rather than just using it as a virtual test bed.

The researchers behind GRIBBOT [28] developed an automation solution for separating chicken fillet from a carcass. Their system consists of a vision solution, a 6-axis robot and a gripper tool for grasping and separating the chicken fillet. They also show how incorporating compliance in the gripper tool can make the solution robust to uncertainties from the vision system. Our system contains the same components and we also use compliance in the gripper to handle uncertainties and variation in the meat products. However, we focus on more general solutions for handling multiple tasks.

In terms of placing the deformable meat pieces, a closely related field is draping operations for cloth. To solve this problem, Balaguer et al. [1] proposed to combine reinforcement learning and learning by demonstration to train a robot system to fold a towel. Other researchers have shown how visual servoing can be used to fold cloth [40]. In our work, we also deal with fairly flat objects where draping operations to some level are necessary to achieve a nice placement. The pork bellies handled in our work are more rigid, which makes them easier to place and thus we can utilize simpler placement strategies. However, the individual products vary more and therefore it is necessary with a placement operation that is robust to the product variation. To achieve this, our work focusses more on determining robust placement actions based on optimization.

In terms of robotic solutions, our main contributions are a novel gripper tool and strategies for grasping and placing the meat based on point clouds from the vision system.

2.3 Numeric Optimization of Robotic Systems

Numeric optimization has been applied to several robotic problems to determine stable solutions based on real world trials [2, 6, 15, 36]. However, limited work has addressed simulation-based optimization of robotic solutions, where the systems are tested in simulation rather than the real world. The advantage of simulation-based optimization is that the number of real world trials can be heavily reduced. Besides speeding up the integration process, this also reduces the chance that real products are damaged during the test phase. When handling meat products, this is especially useful since the meat products have to be changed often to avoid contamination and health hazards. Thus testing in simulation can make the test phase substantially cheaper. Furthermore, it is often easier to set-up experiments and adjust various hardware settings in simulation compared to doing it in the real world, as we demonstrated in [18].

Buch et al. [4] proposed to use simulation-based optimization to determine robotic action parameters for executing a peg-in-hole operation. In their work, they only optimize 2 parameters. Thus they are able to use brute-force like methods to determine a good parameter set. Bodenhagen et al. [3] also rely on simulation-based optimization to tune their action for doing peg-in-hole and laying down operations with deformable objects. Their solutions again rely on only 2 and 3 parameters, and thus they are able to use brute-force like techniques. In our work, we rely on more parameters to define the solutions and thus we focus on optimization techniques that can deal with this in a computationally tractable manner.

Wolniakowski et al. [39] focus on optimizing gripper design in simulation. To achieve this gradient descent based methods are used to determine 11 parameters specifying the gripper fingers. In our work, 12 parameters are optimized, so the scope of the problems are similar. However, we focus on using optimization based on function fitting, in particular "RBFopt" [7], since earlier work [20] indicated this technique is more suitable for this type of optimization problem.

One of the robotic problems that have been optimized based on real world trials is maximizing the walking speed of bipedal robots [6, 15]. In both approaches optimization based on function fitting is used to determine the parameters that result in the fastest robots. Similarly Tesch et al. [36] optimize the speed of a snake-like robot. Again they show optimization based on function fitting have the best performance in terms of quickly optimizing their 7 free parameters.

Our main contributions in the field of parameter tuning is a new use case, where we show simulation-based optimization is suitable for designing robot solutions for handling deformable objects in an industrial setting.

3 Method

Robotic handling of meat is a challenge as few prior assumptions can be made in design. For example, we cannot design towards a specific shape and size as is common in

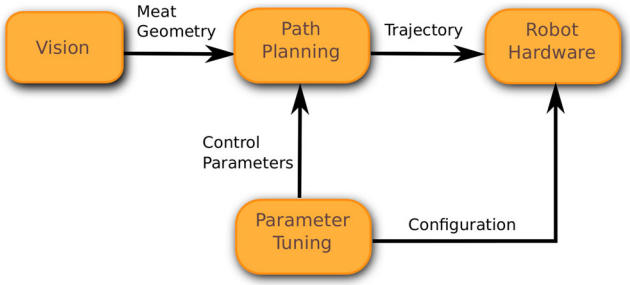


Fig. 2 Diagram of the system architecture. The upper three boxes constitute the runtime system

contemporary robotics. Additional we do not have prior knowledge on the object's pose. As such the exact geometry and the pose must be acquired during the runtime of the system. One way to accomplish this is through 3D vision technology.

Physically moving the object requires adaptable automation. The 6-axis robot arm is ideal for this purpose as it gives us maximum freedom of movement. Furthermore, the robot arm must be equipped with a gripper that is flexible enough to handle the variation and deformation of the meat. The gripper should also be adaptable to different types of meat cuts. Either in runtime or after a short preparation stage.

Picking and placing are not trivial either, as the object of interested should be placed in a specific pose. The deformable nature of the meat pieces makes this need even more pressing. As such our system is equipped with a sophisticated path planning system for determining an appropriate grasp and placement action based on the vision input.

Our system can be roughly broken down into **Vision**, **Gripper** and **Planning** components. However, this alone is not enough as all components contain parameters that must be tuned to a given problem. A large part of the setup time goes to this tuning process. Therefore we have developed a **Simulation** framework, which can handle a huge chunk of the optimization in a virtual environment.

The entire system is illustrated in two flowcharts. The first (Fig. 2) describes the physical system. The diagram also indicates, which parameters are tuned using simulation-based optimization. The second flowchart (Fig. 3) indicates how the system parameters are optimized in simulation. The optimization happens in an iterative procedure, where different software and hardware parameters are tested to determine which produce the best result.

4 Vision

For the robot system to properly locate and handle the meat pieces, it must be supplied with 3D data. By far the most flexible way to achieve this is through vision technology. There has been a huge surge in 3D vision applications due to the wide availability of user friendly real-time scanners such as the Microsoft's Kinect and Intel's Real-Sense. While they are great, they have made a lot of sacrifices to reach real-time performance on a low-cost embedded platform. This means that the accuracy and precision of their 3D data is subpar. For example both versions of the Kinect has an accuracy in the near 1cm range [12].

We instead choose to go another route, by using a similar technology as employed in the above examples, but customized to our needs.

4.1 Structured Light Scanning

Structured light is an active 3D scanning technology that estimates depth via stereo triangulation [11]. The basic idea is the same as with passive stereo vision. By finding the same points projection in a stereo image pair and knowing the relative camera geometry, it is possible to infer the 3D position of that point. The first part is known as the correspondence problem and it can be quite challenging. In passive stereo non-unique and weak texture creates uncertainty which has to be resolved with e.g. statistical priors like spatial smoothness [35].

Instead of relying solely on material appearance, we can project light patterns onto the scene to create artificial texture. By building a certain structure into the projected pattern, the correspondence can be made a lot easier.

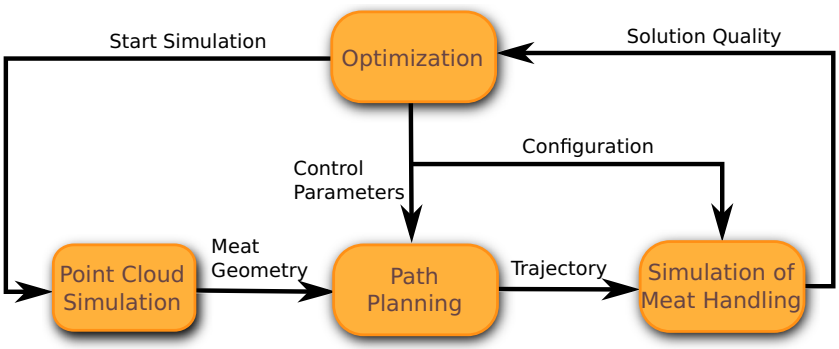


Fig. 3 Diagram of the simulation-based parameter optimization. This process describes how the control parameters and configuration are determined in the parameter tuning block of Fig. 2

Hence the name; structured light. There exist many different encoding strategies ranging from the one-shot, speckle patterns of the Kinect and the Real-Sense to multi-pattern approaches such as Gray Codes [31] and Micro Phase-shifting [13]. First we will go over our hardware setup, afterwards we will discuss the specific structured light method used.

Our scanner consists of three components: two high-definition cameras and a light projector. Depth is estimated via stereo triangulation using the pixel disparities between the camera image pair. This is illustrated in Fig. 4.

In theory, triangulation could also be done between the projector and a camera. However, this requires a projector that has a very well-defined linear gamma curve. Most consumer projectors cannot be used here. By adding a second camera, we ease on the hardware requirement of the projector. This is due to that the projected pattern need no longer be accurately portraited, but that it simply has to be horizontally unique.

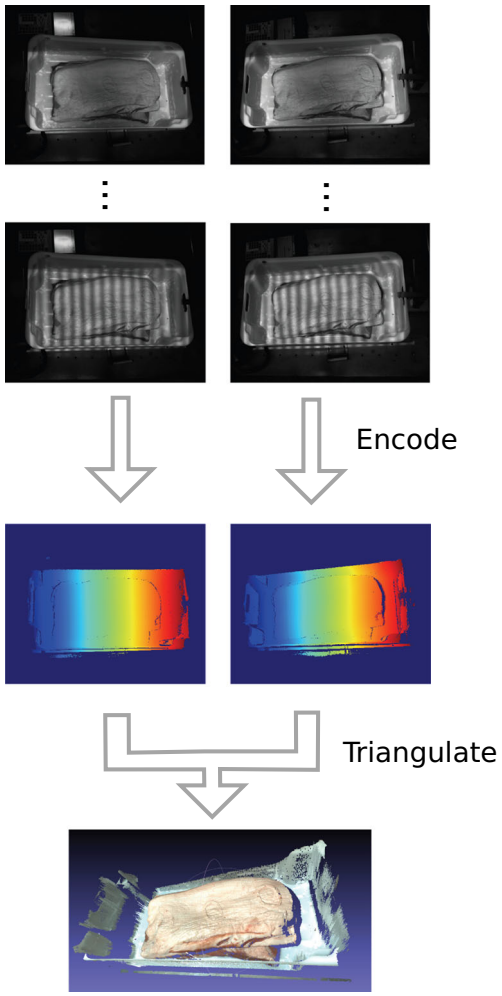


Fig. 4 Illustration of the encoding and triangulation flow of a structured light scan. The first row shows the scene, the second shows the encoded phase as per Eq. 1 and the final shows the triangulated point cloud

As mentioned, we project a series of patterns onto the scene and acquire a series of images from both cameras. Then the idea is to use these patterns to, as the name suggests, encode a continuous phase across the scene. This value can then be used to efficiently solve the correspondence problem. Formally we consider a situation with N projected patterns. Each pixel in each projected pattern should conform to the following spatio-temporal model,

$$I_i(x, y) = \sin\left(2\pi\left[\frac{i}{N} + \frac{\omega \cdot x}{w}\right]\right), \tag{1}$$

where i is the sequence number, ω is the spatial pattern frequency and w is the pattern width. The first term, $\frac{i}{N}$, defines the temporal component of the waveform and the second term, $\frac{\omega \cdot x}{w}$ defines the spatial component. The latter defines a constant, unique phase for each pixel. This is true for both the projected pattern and any image taken of it. We can use the acquired pattern series to estimate $\frac{\omega \cdot x}{w}$ for each pixel. Figure 4 illustrates the overall process in the method.

We refer the interested to [11] for specific implementation details.

4.2 Segmentation

Of course, a point cloud generated by structured light is not particularly useful in itself. It must be segmented into meaningful parts before the information can be utilized in path planning. Specifically, we want each meat piece as separate segments. We accomplish this via a modified version of the region growing segmentation algorithm available in Point Cloud Lib [32, 33]. Our version is shown in Algorithm 1. It grows a region from a point of low curvature and terminates at high curvature and change in normal angle. The outermost loop of Algorithm 1 repeats until all points in the point cloud have been exhausted. For each iteration the point with lowest curvature $p_{\min}$ is removed from S. If it has not been assigned a region (indicated by A) a new region growth is initialized from $p_{\min}$ by creating a seed list S_c and a region list R_c. An inner loop over S_c is then initialized. Here neighboring points are cycled over for each point in S_c referred to as p_i. Each point in the neighborhood p_j is then tested whether it has been assigned to another region. If not, the algorithm continues to test the angle between normals of p_i and p_j. If it is above a certain threshold the point is discarded. Otherwise, it is added to region R_c and registered as being assigned in A. Finally, the curvature of p_j is tested. If it is below a certain threshold then p_j is added to S_c so the growth can be continued from p_j.

Algorithm 1: Segmentation via image space region growth.

Data:

$P =$ organized point cloud,
$N =$ organized point normals,
$C =$ organized point curvature,
$c_t =$ curvature threshold,
$\theta_t =$ angle threshold.

Result:

$R =$ list of segmented regions.

$\{w, h\} \leftarrow \text{size}(P)$;
$R \leftarrow \emptyset$;
$A \leftarrow \text{zeros}(w, h)$;
$S \leftarrow$ set of all points in P;
Sort S by ascending order of curvature;
while $S \neq \emptyset$ **do**
  $p_{\min} = (x, y) \leftarrow$ head of S;
  $S \leftarrow S \setminus p_{\min}$;
  if $A(x, y) = 1$ **then**
    continue;
  end
  $S_c \leftarrow \{p_{\min}\}$;
  $R_c \leftarrow \emptyset$;
  while $S_c \neq \emptyset$ **do**
    $p_i \leftarrow$ head of S_c;
    $S_c \leftarrow S_c \setminus p_i$;
    $R_c \leftarrow R_c \cup p_i$;
    $B \leftarrow$ 8-neighbors of p_i;
    for p_j *in* B **do**
      $\{x_j, y_j\} \leftarrow p_j$;
      $\{x_i, y_i\} \leftarrow p_i$;
      if $A(x_j, y_j) = 1$ **then**
        continue;
      end
      $a \leftarrow N(x_j, y_j) \cdot N(x_i, y_i)$;
      if $a < \cos\theta_t$ **then**
        continue;
      end
      $R_c \leftarrow R_c \cup p_j$;
      $A(x_j, y_j) \leftarrow 1$;
      if $c(x_j, y_j) < c_t$ **then**
        $S_c \leftarrow S_c \cup p_j$;
      end
    end
  end
  $R \leftarrow R \cup R_c$;
end

The main difference in our algorithm and the one available in Point Cloud Lib [32] is that ours is tailored specifically towards organized point clouds, meaning point clouds that are given in a 2D grid. This is the typical output format of e.g. the Kinect and our structured light scanner. The main performance limiter for a generic point cloud is the search for neighbors. This, along with various control logic, can be greatly sped up by exploiting the grid location of a given point. On an Intel Core i7-4610M it segments a point cloud of size 675x540 in 100ms-150ms.

After segmenting the point cloud, we must determine which is the next meat piece that should be handled. This is achieved by locating the five largest point clouds and selecting the top most point cloud of these. The process in its entirety is shown in Fig. 5.

5 Pick and Place Operations

In this work, we focus on a fairly general pick and place operation where multiple meat pieces are placed in a box and have to be moved to a conveyor belt. Furthermore, the meat should be placed stretched out such that it is ready for post-processing. Automating this task is a challenge as the meat is deformable and each cut varies significantly. To solve the task, two components are required: First, a hardware solution has to be designed to move the meat. Secondly, a mechanical motion for lifting and placing the meat has to be generated.

As discussed in Section 1, a 6-axis robot with a flexible and adaptable suction based gripper tool is used to lift the meat. The gripper attached to the robot is discussed in Section 5.1.

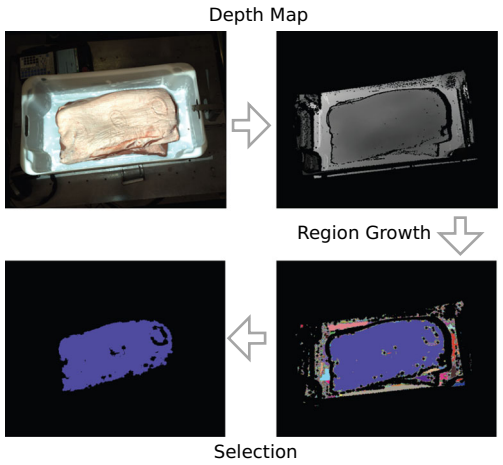


Fig. 5 Flow of the segmentation and selection process of the observed meat pieces

Table 1 Control parameter bounds used during optimization of the placement action

	Gripper	Rolling grasp				Placement			
	d	d_{ideal}	w	gl	gr	d_{x1}, d_{y1}, d_{z1}	θ	b	d_{x2}, d_{y2}
Min	130mm	20mm	0	100mm	15°	0mm	0°	0	0mm
Max	170mm	50mm	1	200mm	25°	100mm	20°	2	100mm

Besides designing a hardware solution, a robotic motion for lifting and placing the meat also has to be developed. These motions are discussed in detail in Sections 5.2 and 5.3 respectively.

Both these motion strategies are determined based on the vision input derived as discussed in Section 4. Furthermore as one might expect, the parameterization of the robotic hardware and motions contain several free parameters, which have to be determined. In this work, these parameters are determined using simulation-based optimization as discussed in Sections 6 and 7. A full list of the parameters are given in Table 1 and they are explained in detail in this section.

5.1 The Gripper

When designing the gripper tool, one of the key challenges is the high variation between each pick. Multiple aspects contribute to this variation. First of all, the size, shape and deformability of the meat pieces vary even within the same type of meat cut. Furthermore, the placement and deformed state also vary as each meat piece is placed differently in the box. Besides simply being flexible enough to handle one type of meat cut, the gripper should also be adaptable, such that it can be adjusted to handle different cuts. The gripper design used for addressing these challenges can be seen in Fig. 6 and the real gripper is shown in Fig. 7.

To grasp the meat, the gripper relies on suction cups, similar to [19]. The suction cups are flexible, so they can adapt to the local surface variation of the meat pieces. This is necessary to ensure that no air leaks into the vacuum chamber which would result in the suction cup dropping the meat. However, the local surface adaptation is not enough to deal with the larger variations that can occur across an entire meat piece. To address this, the suction cups are placed at the end of air pistons, which act as passive components much like if they were replaced with one-dimensional springs. These air pistons can be compressed a lot more than the suction cups, and enable the tool to adapt to larger deformation.

Besides being flexible, the gripper should also be adaptable such that it can grasp meat cuts of different sizes. To achieve this the distance between the suction cups, d, can be changed to match a particular meat cut. In this work, d is considered a control parameter which is optimized in simulation. A deeper discussion of the gripper design is given in [19].

5.2 The Rolling Grasp

The goal of the grasping strategy is to lift the meat robustly. A key challenge here is that a vacuum can form between the meat piece that is to be lifted and the piece below. If this vacuum becomes too strong, it will result in the grasp failing because the gripper lifts the two pieces sticking together.

To address this challenge a rolling lift was designed where the suction cups are placed close to the edge of the meat and lifted in a rolling motion, as illustrated in Fig. 9. This allows air to flow under the meat which increases the

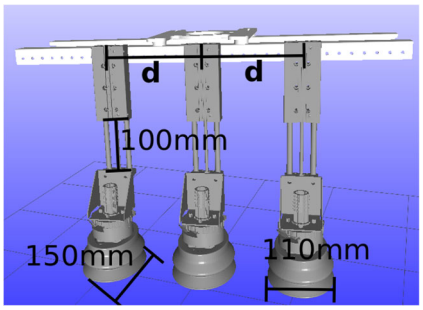


Fig. 6 A cad model of the gripper tool

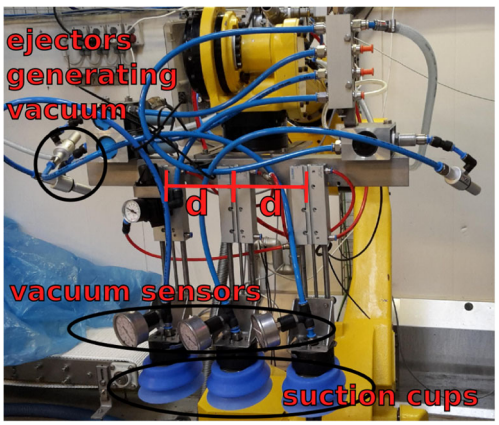


Fig. 7 The suction based gripper relies on ejectors for generating the vacuum, and it contains 3 sensors for measuring the pressure levels at each suction cup

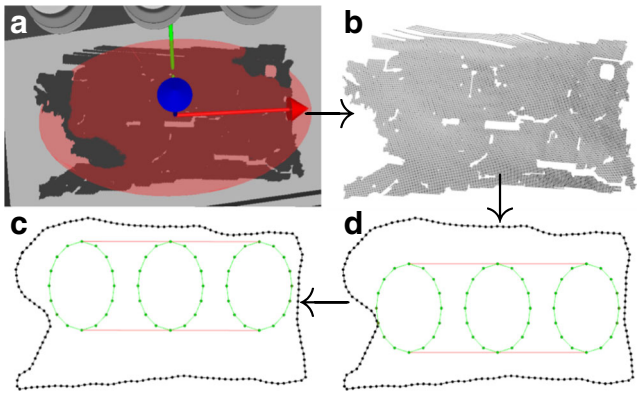


Fig. 8 Placement of the suction cups. **a** A PCA is applied to the grey 3D point cloud. The red and transparent ellipsoid represents the eigenvector and eigenvalues of the PCA. The frame is the PCA frame. **b** The point cloud is projected onto the PCA frame to generate a 2D point cloud. **c** The black dots show the concave hull of the 2D point cloud that is used to represent the meat edge. The green dots represent the initial suction cup placement aligned with the PCA frame. **d** The final suction cup placement is determined by minimize a regret score

chance that the meat is separated from the piece below. The benefit of using this fairly complex grasp strategy over a simpler approach is demonstrated in [19].

The grasp is generated in two stages: First, an acceptable suction cup placement is determined based on the segmented point cloud of the meat piece, discussed in Section 4. This process is illustrated in Fig. 8. Then a robotic trajectory is defined to move the suction cups to the determined positions and lift the meat piece in a rolling motion. This motion is illustrated in Fig. 9.

In order to place the suction cups close to the edge, the edge of the meat has to be determined. To achieve this, a PCA of the segmented point cloud is conducted (Fig. 8a). Then the point cloud is projected onto the x,y-plane of the PCA frame (Fig. 8b). Finally, the edge can be determined as a concave hull of the projected 2D points (Fig. 8c). This is achieved using the concave hull algorithm from PCL [32]. After the edge is determined, it is re-sampled to a resolution of 10mm to have a uniformly sampled edge model.

The next step is to determine the placement of the suction cups based on the edge model. This placement has to satisfy three conditions. First, the suction cups should be placed within the meat. Secondly, the suction cups should be placed close to the edge. Lastly, a large part of the meat edge should be close to the suction cups. To determine a placement that satisfies these conditions, we pose the problem as a minimization problem where a regret score is minimized. The regret score, R, captures how well the placement satisfies the conditions and it consists of two parts R_{cups} and R_{meat}. R_{cups} ensures that the suction cups are placed close to the edge while still being inside the meat. R_{meat} ensures that a large part of the meat edge is close to the suction cups. For the particular case where three oval suction cups are placed on a rectangular meat piece: R_{cups} favors that the suction cups are placed close to the long edge of the meat, while R_{meat} favors that the suction cups are placed close to the corners of the meat piece.

To control how close the suction cups and the meat edge should be, the control parameter d_{ideal} is introduced. d_{ideal} represents the ideal distance between the suction cups and the meat edge and it should be determined through simulation-based optimization. The regret score and the two subcomponents are given in Eqs. 2, 3 and 4.

$$R_{\text{cups}} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}(min(\|\mathbf{s}_i - \mathbf{P}\|) - d_{\text{ideal}})^2, \\ \qquad\qquad \text{all } \mathbf{s}_i \text{ are inside the meat} \\ 1.0, \qquad\qquad\qquad \text{otherwise} \end{cases} \tag{2}$$

$$R_{\text{meat}} = \begin{cases} \frac{1}{M}\sum_{j=1}^{M}\sqrt{|min(\|\mathbf{p}_j - \mathbf{S}\|) - d_{\text{ideal}}|}, \\ \qquad\qquad \text{all } \mathbf{p}_j \text{ are outside the suction cups} \\ 0, \qquad\qquad\qquad \text{otherwise} \end{cases} \tag{3}$$

$$R = w \cdot R_{\text{cups}} + (1 - w) \cdot R_{\text{meat}}^4 \tag{4}$$

where $\mathbf{s}_i$ is a point on the suction cups. Each suction cup contains 16 points placed on the periphery, as illustrated by green dots in Fig. 8c and d. $\mathbf{P}$ is the meat edge. To determine R_{cups} the smallest distances from the suction cup points to the meat edge are squared, to favor that all the points on the suction cups are close to the edge.

$\mathbf{p}_j$ is a point on the meat edge. $\mathbf{S}$ is the suction cup edges. To determine R_{meat}, the square root of the smallest distances from points on the meat edge to the suction cups are used. This is done to ensure outliers do not dominate the score since some edge points will be far away from the suction cups. This can be seen in Fig. 8d, where there are many points on the meat edge (black dots) that are far away from the suction cups. This way the score favors many inliers, over being close to every point.

Finally, the regret score is determined based on a tradeoff, w, between the two subcomponents. This tradeoff is a control parameter which should be optimized in simulation.

To determine a good suction cup placement based on the regret score, the minimization algorithm coordinate descent [26] is used. This algorithm moves the suction cups around in the 2D-plane, to find the placement with the lowest regret.

After the suction cup placement is determined, the next step is to determine the actual robot motion. The purpose of this motion is to lift the meat while avoiding a vacuum forming below it. This is achieved by lifting the meat in a rolling motion, such that air can flow in and separate the meat from the surface below. The motion is produced by

the robot moving through three frames, which is illustrated in Fig. 9. The first frame is the grasp frame, GF, and it describes where the suction cups should be placed to grasp the meat. After the robot reaches the grasp frame the suction cups are activated to initiate the lift. Then the robot moves to the via frame, VF, which ensures the meat is lifted in a rolling motion. Lastly, the robot moves to the lift frame, LF, which ensures the meat is lifted well above the box.

The three frames are determined based on two control parameters named gr and gl, which should be optimized in simulation. Both gr and gl are illustrated in Fig. 9. The grasp frame is determined by reprojecting the optimal suction cup placement back into the 3D-world. The via frame is determined by rotating the grasp frame around the y-axis of the frame, by an angle specified by gr. Lastly, the lift frame is determined by translating the grasp frame in the z-direction by a distance specified by gl.

5.3 The Placement Operation

The goal of the placement strategy is to place the meat, such that it can be wrapped in folio by a wrapping station. To achieve this, the meat should be placed stretched out on the conveyor belt with a flat front facing the wrapping station, as illustrated in Fig. 10. This operation is fairly specific, but the placement criteria itself is common in the meat sector. E.g. it is a requirement if the meat is to be placed in boxes and for various cutting operations.

To enable the robot to place the meat in this manner, the placement strategy is designed to stretch the meat as it collides with the conveyor belt. This stretching is achieved by moving the gripper through two frames as it moves towards the final position over the conveyor belt. As the tool reaches these frames, the meat collides with the conveyor belt which stretches it. If the frames are picked reasonably, the meat is more likely to be placed in the desired fashion. When the gripper reaches the final position, the suction cups release the meat and the robot moves away. The entire placement strategy is illustrated in Fig. 11, and especially 11c shows how the collision with the conveyor belt can stretch the meat.

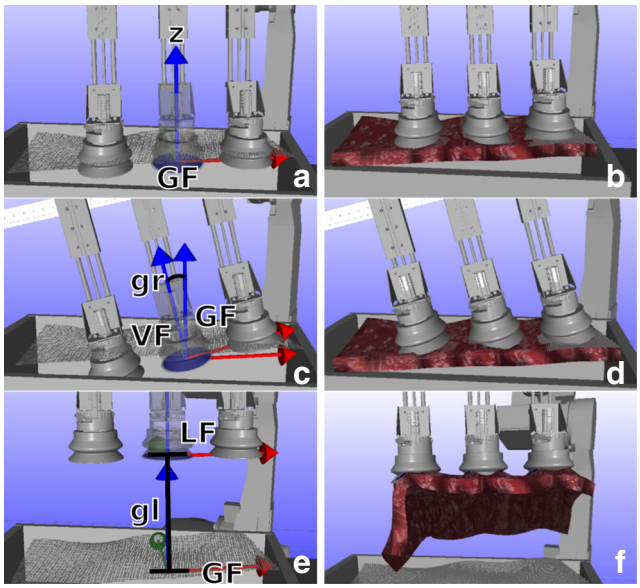


Fig. 9 Rolling lift - planning and simulation. **a**, **b** show how the tool is aligned with the grasp frame after the grasp frame is mapped to the 3D world. **c**, **d**, **e**, **f** illustrate the rolling motion that is used to lift the meat. The via frame, VF, and the lift frame, LF, are used to define the motion as indicated

Fig. 10 Human meat placement. The meat is placed on the conveyor belt with a flat front facing the wrapping station

The three frames that define the robotic motion are named placement frame, PF, first approach frame, $AF1$, and second approach frame $AF2$. The placement frame specifies where the suction cups should be placed when the meat is released.

At the end of the rolling grasp (Fig. 9f) it can be seen that a large part of the meat hangs down to the left and at the back of the suction cups. To ensure this is stretched out the first approach frame was introduced. The height of the frame is chosen, such that the corner of the meat to the left and at the back roughly touch the conveyor belt. Furthermore, the frame is moved slightly to the left and further back to ensure the meat is stretched as the robot moves towards the placement frame.

Initial trials using only the first approach frame and the placement frame resulted in the meat being twisted during the placement. The result was similar to the placement shown in Fig. 11c. This twist was corrected by introducing the second approach frame. This frame ensures the meat is dragged a bit too far, such that when it moves back to the placement frame the twist will be reduced as illustrated in Fig. 11d.

Both of the approach frames are dependent on several control parameters, which should be optimized in simulation to ensure a robust placement operation. These parameters are d_{x1}, d_{y1}, d_{z1}, d_{x2}, d_{z2}, θ and b. All parameters

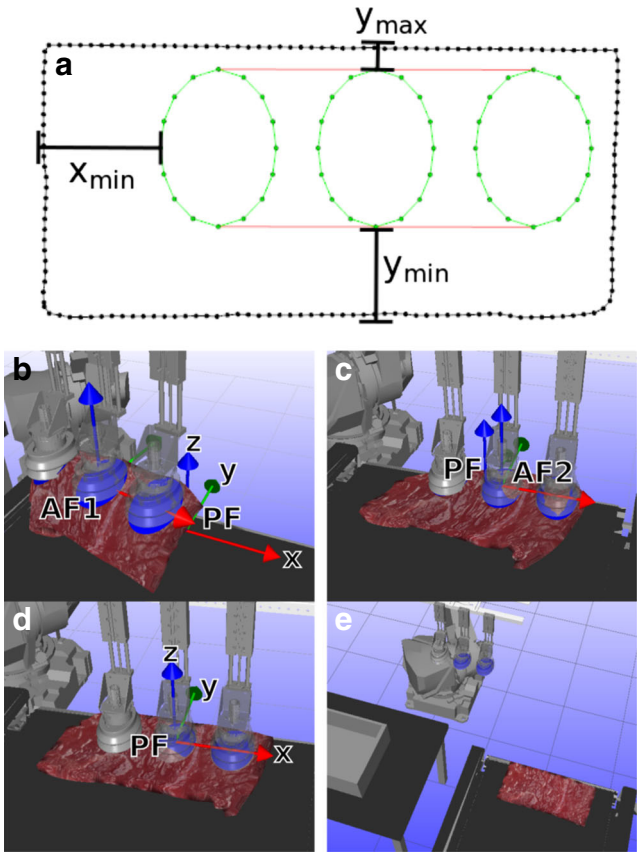


Fig. 11 Placement Action. **a** illustrates the 2D placement of the suction cups used during the grasp. The distances x_{min}, y_{min} and y_{max} is used to determine the placement action. **b** shows the tool moving to the first approach frame, $AF1$, and **c** shows it moving to the second approach frame, $AF2$. **d** shows the tool as it reaches the placement frame, PF, and finally in **e** the vacuum is turned off and the robot is moved away

except b represent different offsets to the translation and rotations of the approach frames. b specify how much the meat hanging down at the edges of the suction cups should be considered in the translation of the first approach frame.

Mathematically the first approach frame is defined as the placement frame translated by $(x_{AF1}, y_{AF1}, z_{AF1})$ and rotated around the z-axis by θ_{AF1}, these values are given in Eqs. 5, 6, 7 and 8.

$$x_{AF1} = -(x_{min} \cdot b + d_{x1}) \tag{5}$$

$$z_{AF1} = x_{min} \cdot b + d_{z1} \tag{6}$$

$$y_{AF1} = \begin{cases} -(y_{min} \cdot b + d_{y1}), & y_{min} > y_{max} \\ y_{max} \cdot b + d_{y1}, & \text{otherwise} \end{cases} \tag{7}$$

$$\theta_{AF1} = \begin{cases} -\theta, & y_{min} > y_{max} \\ \theta, & \text{otherwise} \end{cases} \tag{8}$$

where x_{min}, y_{min} and y_{max} are distances between the suction cups and the edge of the meat. These distances are illustrated in Fig. 11a, and they are used to ensure that the stretching of the meat is dependent on how much meat is hanging down at the edges of the suction cups.

The z_{AF1} translation ensures that the meat roughly touches the conveyor belt. The x_{AF1} translation ensures that the meat hanging down to the left of the suction cups is stretched. The y_{AF1} translation ensures the meat is stretched in the y direction as well. Whether the meat should be stretched in the positive or negative y direction depends on where the suction cups are placed on the meat. Finally, initial trials indicated that the meat is slightly rotated when the y translation is introduced. Therefore the rotation θ_{AF1} was added to reduce the other rotation.

The second approach frame, $AF2$, is defined as the placement frame translated $(d_{x2}, 0, d_{z2})$. The x translation is introduced to ensure the meat moves too far, such that it can move back to reduce the twist of the meat (Fig. 11c). The z translation is introduced to ensure the meat is not pushed too hard into the conveyor belt.

6 Simulation

To optimize the robot system in simulation, the first step is to construct a simulation framework for modeling robotic handling of the meat pieces. A central part of this framework is the deformation model for the meat pieces. In this work, a mass-spring model is used.

This model consists of several particles, which motion is constrained by various springs placed between them. Throughout this paper the particles in the mass-spring model is refered to as meat particles. The model is described in detail in [17]. As discussed in [17], a spring-model was chosen over more complex finite element models, similarly to [27, 29]. The main reason for this is that spring-models tend to to be less computationally expensive. This is favorable since many simulations have to be conducted both to evaluate the robustness of potential solutions and to optimize the overall solution.

In the real scenario, several steps are taken. First, the meat is dropped in a box. Secondly, a scanner generates a segmented point cloud of the top meat piece, as discussed in Section 4. This point cloud is used to determine a robotic pick and place action for moving the meat, as discussed in Section 5. Then the robot pushes the unactivated suction cups into the meat. Next, the suction cups are activated, such that the robot can lift the meat. Finally, the meat is placed on the conveyor belt. All these steps require modeling several interactions, these models are described in Sections 6.1 to 6.5.

To simulate the full process, the meat piece is first placed above the box. It is then translated by $(M_x, M_y, 0)$ and rotated by M_R around the z-axis to randomize the initial

position. Then it is dropped such that it falls into the box, as described in Section 6.1. After the meat settles, a point cloud of the meat piece is rendered as discussed in Section 6.5. This point cloud is used to generate the robotic pick and place action for moving the meat to the conveyor belt.

After the action is generated, the robotic motion is simulated. First, the suction cups are moved down to the meat, as discussed in Section 6.2. Then the suction cups are activated and lifted according to the grasp strategy, as discussed in Section 6.3. Then the suction cups move the meat to the conveyor belt where it is placed, again based on the model described in Section 6.2. After it is placed the suction cups are lifted, and then the simulation ends. Images from the simulation can be seen in Figs. 9 and 11. Throughout the simulation, the motion of the suction cups is determined based on a model described in Section 6.4.

6.1 Initial Box Constraint

The purpose of this constraint is to model the interaction between the meat piece and the box that the meat arrives in. This constraint should also capture that there might be several meat pieces below the top piece.

To model the uneven surface of a box full of meat a thin plate spline (TPS) [8] is used. This spline guarantees a smooth surface, and yet it can be highly randomized to capture many different initial conditions. The initial box surface, with a meat piece laying on it, is illustrated in Fig. 12.

The spline is defined based on 4×7 knot points, which determine the shape of the surface. The knot points are placed on a regular grid, which matches the shape of the box with the meat pieces. The height of the knot points is randomized, to roughly model that the meat pieces below are placed randomly.

If a meat particle moves through the thin plate spline, it is considered in contact with the initial box constraint. When this happens, the particles motion is fixed to the point of contact.

This constraint should capture two phenomena. The first is that the meat can not move through the surface of the box. The second is that a vacuum can form between two meat pieces placed in the box.

To model both these aspects, the meat particle is fixed to the point where it comes in contact with the surface. To move it two conditions have to be satisfied. The first condition is that air can flow below the meat particle. This is modeled by requiring that at least one of the neighboring meat particles is free of the initial box constraint. The second condition is that the meat can not move further into the surface. This is modeled by requiring that the meat particle is lifted.

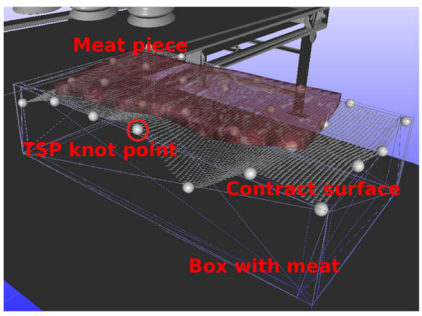


Fig. 12 The meat piece laying on the initial contact surface. The grey point cloud represents the thin plate spline, and the big grey points represent the knot points determining the shape. Furthermore, the meat piece is transparent and the box with the meat is represented as lines, to make the knot points of the thin plate spline fully visible

6.2 Planar Constraint

The purpose of this constraint is to model the interactions between the meat piece and a planar surface. In this work, the un-activated suction cups and the conveyor belt is modeled with planar constraints.

The constraint has a planar surface and a 2D boundary shape, for the suction cups the shape is an ellipse, and for the conveyor belt it is a rectangle. In case a meat particle comes in contact with the constraint, a contact point is added where the particle collides with the surface. In case the particle moves further into the surface, it is moved back to the contact point. In case it moves away from the surface the constraint is removed. When using the constraint to model a suction cup, the contact point moves along the surface of the suction cup. The constraint in action is illustrated in Fig. 13, where it keeps the meat from falling through the conveyor belt.

6.3 Vacuum Constraint

The purpose of this constraint is to model the interactions between the meat and the activated suction cups.

When the suction cups are activated in the real world, the meat is quickly attached to the suction cups. To model this

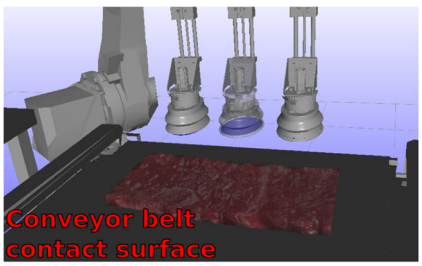


Fig. 13 A planar constraint is used to ensure the meat does not fall through the conveyor belt

in simulation, a contact volume is used to determine which meat particles are in contact with the suction cups. This volume is an elliptic cylinder that is formed by the surface of the suction cup ±5mm. When a suction cup is activated, all the meat particles within the contact volume is projected onto the surface of the suction cup. The meat particles are then fixed to these projected points until the suction cups are deactivated. The constraint in action can be seen in Fig. 14, where it constrains meat particles to the blue suction cup surface.

6.4 Suction Cups

Besides modeling the interaction between the meat and the suction cups, the motion of the suction cups themselves also has to be modeled. The model should capture the linear motion of the air pistons placed above the suction cups, and the local adaptation of the suction cups themselves.

This is achieved by modeling a suction cup as a planar elliptical mass placed at the end of a linear and angular spring, as illustrated in Fig. 14. The other side of the linear spring is attached to the gripper tool, which position is kinematically determined based on the robotic motion. The forces and torques affecting a suction cup are determined based on the meat particles in contact with the suction cup.

6.5 Point Cloud Rendering

Besides modeling the mechanical interactions between the meat piece and its surroundings, a point cloud renderer was also introduced. This step is needed to generate input data for the pick and place strategy, which determine the robot motion based on a segmented point cloud of the meat. To ensure the meat piece is segmented, a new scene only containing the meat piece is generated and then the point cloud is captured in this scene. The RobWork [10] point cloud renderer was used to generate the point cloud, and the final result is illustrated in Fig. 15.

7 Optimization in Simulation

To determine an action that places the meat pieces stretched out on a table, we propose to use simulation-based optimization. The simulation used is discussed in Section 6. The strategy used to place the meat is based on various control parameters, and it is these parameters that should be optimized. Furthermore, the use case contains many uncertain parameters, that should be analyzed to ensure the performance of the solution is stable. Both the control parameters and the uncertain parameters are discussed in Section 7.1.

In order to optimize the control parameters, it is necessary to quantify the quality of a placement based on the simulations. This score should favor actions that place the meat stretched out on a table. The score for achieving this is discussed in Section 7.2.

Finally, the process for determining a robust solution based on numeric optimization is discussed in Section 7.3. This process is applied to two different cases. In the first the robot has to pick pork bellies and in the second it has to pick pork loins.

7.1 Free Parameters and Uncertainties

Several control parameters that are crucial for generating a stable placement action were selected for optimization. All the parameters are listed in Table 1, the first parameter specifies the gripper design, the next 4 specify the grasp action and the last 7 specify the placement action. Besides just listing the parameters the table also shows the parameter bounds used during optimization of the entire pick and place action. These bounds are selected based on hardware

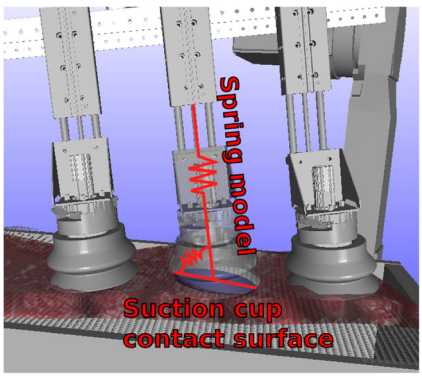


Fig. 14 The spring based suction cup model ensures the blue surface of the suction cup aligns with the meat during grasping. The middle suction cup and the meat are transparent to better visualize the blue surface of the middle suction cup

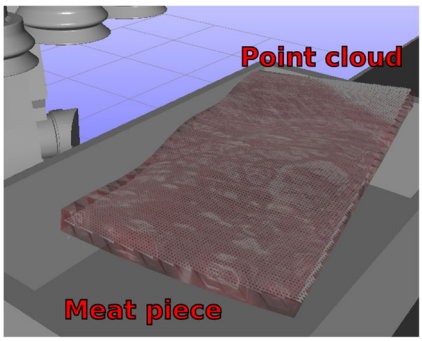


Fig. 15 The simulation of the grey point cloud is done using RobWork [10]. The meat piece is transparent such that the entire point cloud can be seen. Furthermore, the point cloud is slightly translated and rotated to model the effect of uncertainties in the vision system

Table 2 Parameter bounds for the uncertain values of the pork belly, which is used to analyze the robustness of the solutions

	Meat cutout					Vision		Contact surface	
	M	S_{def}	S_{size}	$M_{x,y}$	M_R	$V_{x,y,z}$	$V_{R,P,Y}$	S_{offset}	d_{tps}x28
Min	3.5kg	0.9	0.95	−50mm	−10°	−10mm	−3°	0mm	0.0
Max	5.5kg	1.1	1.05	50mm	10°	10mm	3°	200mm	1.0

limitations and to ensure that the meat pieces are placed on the conveyor belt. All the control parameters have been described in detail in Section 5.

Besides the control parameters, the system also contains a lot of uncertain parameters, such as meat size and deformability. To ensure the solution can cope with variation in these parameters, the tested actions should be simulated with different perturbations of the uncertain parameters. To achieve this the first step was to determine the most crucial uncertainties. These are listed in Table 2. Furthermore, the bounds the parameters can occur within should be estimated. These bounds are based on data from the production lines at Danish Crown and the values are given in Table 2.

The first 6 uncertain parameters are introduced to capture the variation between the meat pieces and how they are placed in the boxes. M is the weight of the meat piece. S_{def} is a deformability parameter (Table 3). This parameter model the variation in the deformability of the meat pieces. In reality, this variation occurs due to multiple factors, such as variation in the thickness, fat content and temperature of the meat. In the simulation, the variation is modeled by a scalar multiplied to all the spring constants in the meat model. The base spring constants are chosen to make the simulated meat deform similarly to the real meat pieces, the spring model and the constants are discussed in more detail in [17]. S_{size} is a scaling factor multiplied to the base size of the meat piece, for the pork belly this size is 525 × 250 × 20mm and for the pork loin it is 550 × 120 × 75mm. M_x and M_y are perturbations of the meat piece in the x and y-direction before it is dropped into the box in the simulations. M_R specify how much the meat is rotated around the z-axis before it is dropped.

The following 6 parameters are included to model imperfections in the camera-robot calibration and other uncertainties introduced by the vision system. V_x, V_y and V_z specify perturbations in the x, y and z directions of the point cloud of the meat after it is dropped into the box. V_R, V_P and V_Y specify a roll, pitch and yaw perturbation to the rotation of the point cloud.

The last uncertain parameters are included to model the variation of the box the meat is dropped into. This variation occurs because there can be between 0 and 8 meat pieces below the top piece that is to be grasped. This surface is uneven and in the simulation, it is modeled by a thin plate spline. This spline is specified based on 29 uncertain parameters. First S_{offset} specify the maximum height of any knot point in the thin plate spline. The other 28 d_{tps} parameters are used to specify the height of the individual 28 knot points, while ensuring the points are never placed below the box.

7.2 The Objective Score

To use numeric optimization, an objective score has to be defined. This score should capture the quality of any given set of control parameters, such that the optimal pick and place action can be distinguished from poor actions. In this work, the objective score is determined based on an automated analysis of the simulations. In particular, it is determined by analyzing each meat particle throughout the simulation, which is discussed in Section 6.

To capture the quality of a solution the score should address three different issues. First, it should favor actions resulting in the meat being stretched out on the table. Secondly, it should favor actions where the orientation of the meat matches the desired orientation. Lastly, it should favor actions where the internal forces in the meat are limited, to ensure the meat is not damaged in the operation. These issues are addressed by constructing the final objective score from three different scores.

The first two scores ensure that the rotation and deformation of the meat piece match the desired rotation and deformation after it is placed on the conveyor belt. To

Table 3 Parameter bounds for the uncertain values of the pork loin, which is used to analyze the robustness of the solutions

	Meat cutout					Vision		Contact surface	
	M	S_{def}	S_{size}	$M_{x,y}$	M_R	$V_{x,y,z}$	$V_{R,P,Y}$	S_{offset}	d_{tps}x28
Min	2.0kg	0.9	0.9	−50mm	−10°	−10mm	−3°	0mm	0.0
Max	3.5kg	1.1	1.1	50mm	10°	10mm	3°	200mm	1.0

determine these scores, the first step is to determine the pose of the meat piece after it is moved to the conveyor belt. This is done by using the Kabsch algorithm [21] between the point set representing the desired meat placement and the point set representing the meat piece in the simulation. This returns a pose transformation from the desired point set to the actual point set in the simulation. The rotation from the pose transformation is then used as the rotational error, E_{rotation}. This error is converted to the **rotation objective** through Eq. 9.

$$Q_{\text{rotation}} = \begin{cases} 0 & \text{if } E_{\text{rotation}} > 30° \\ 1 - \frac{E_{\text{rotation}}}{30°} & \text{otherwise} \end{cases} \tag{9}$$

To determine the **deformation objective**, the desired point set is moved onto the final point set using the pose transformation, and then the RMS error between the two point sets are determined. This score is used as the deformation error, $E_{\text{deformation}}$, which is converted into an objective score through Eq. 10.

$$Q_{\text{deformation}} = \begin{cases} 0 & \text{if } E_{\text{deformation}} > 50\text{mm} \\ 1 - \frac{E_{\text{deformation}}}{50\text{mm}} & \text{otherwise} \end{cases} \tag{10}$$

The last objective score is the **force objective**. This score favors solutions that produce small internal forces inside the meat pieces. This score is based on the maximal force exerted on any meat particle throughout the simulation. The maximal force, F_{max}, is converted into an objective score through Eq. 11.

$$Q_{\text{force}} = \frac{6.0\text{N}}{F_{\text{max}}} \tag{11}$$

Finally, all the objective scores are combined into one score, Q, using the geometric mean as shown in Eq. 12. The geometric mean was chosen since it favors solutions where all the objective scores are high. Furthermore, the partial objective scores are all designed to be between 0 and 1, and thus the combined score will also be in this interval. For more detail, on the objective scores, we refer to [17].

$$Q = \sqrt[3]{Q_{\text{rotation}} \cdot Q_{\text{deformation}} \cdot Q_{\text{force}}} \tag{12}$$

7.3 Numeric Optimization

In bounded global numeric optimization, the idea is to determine the parameter set resulting in the highest function evaluation for a multi-dimensional function. This can be expressed by Eq. 13.

$$\mathbf{x}_{opt} = \underset{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}_{min} \leqq \mathbf{x} \leqq \mathbf{x}_{max}}{\text{argmax}} \quad f(\mathbf{x}) \tag{13}$$

In this work, $\mathbf{x}$ is the control parameters that define the pick and place action and $\mathbf{x}_{opt}$ define the best action. $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the bounds which the control parameters should be optimized within. f is based on the objective score, Q, which is calculated in the simulations. To ensure f favors solutions that are robust to the uncertain parameters, it is determined based on multiple simulations with different uncertain parameter perturbations according to Eq. 14.

$$f(\mathbf{x}) = \bar{Q} - 2 \cdot \text{SD}(Q) \tag{14}$$

where $\bar{Q}$ is the average objective score based on multiple simulations. $\text{SD}(Q)$ is the standard deviation of the objective scores.

When computing the score, the variation in Q is achieved by varying the uncertain parameters of the simulation uniformly within the uncertainty bounds. Equation 14 is based on work presented in [18]. In [18], the equation is demonstrated to be effective at determining solutions that work well, even when tested for different perturbations of the uncertain parameters in simulation.

Several tools exist for solving the maximization or optimization problem. In previous work, [20], we showed that *RBFopt* is a powerful optimization algorithm for robotic meat handling and other robotic use-cases where the solutions should be robust to various uncertainties in the system. Thus in this work, we use *RBFopt* to optimize the parameters.

During the optimization, it is infeasible to run a substantial amount of simulations for each parameter set. Therefore, as verified in [20], we propose to do multiple optimization runs where each parameter set is evaluated based on a few simulations. Then for each optimization run, the best parameter set are thoroughly evaluated to determine the very best set. During the optimization, we evaluate the parameter set in simulation based on 10 different perturbations of the uncertain parameters. Furthermore, we do 10 optimization runs with 100 iterations each. After the 10 best parameter sets are determined we evaluate them based on 1000 different perturbations of the uncertain parameters to determine the best parameter set, which is then used as the final solution. This optimization process is done for both use cases, to determine case-specific solutions for both cases.

For the pork belly case, the optimization process is illustrated in Fig. 16. In Fig. 16a, f is plotted throughout the iterations of the optimization runs. Here it can be seen that the objective score increases substantially throughout the optimization. This shows that the pick and place action improve substantially as better and better control parameters are tested. In Fig. 16b, the final solutions of the individual optimization runs are compared, in order to select the very best solution. This solution is at index 6, where f evaluated based on 1000 simulations result in 0.892. The two graphs in Fig. 16b represents solution qualities based on 10 and 1000 evaluations. Due to the similarity between the two graphs, it can be seen that the objective scores based on 10 and

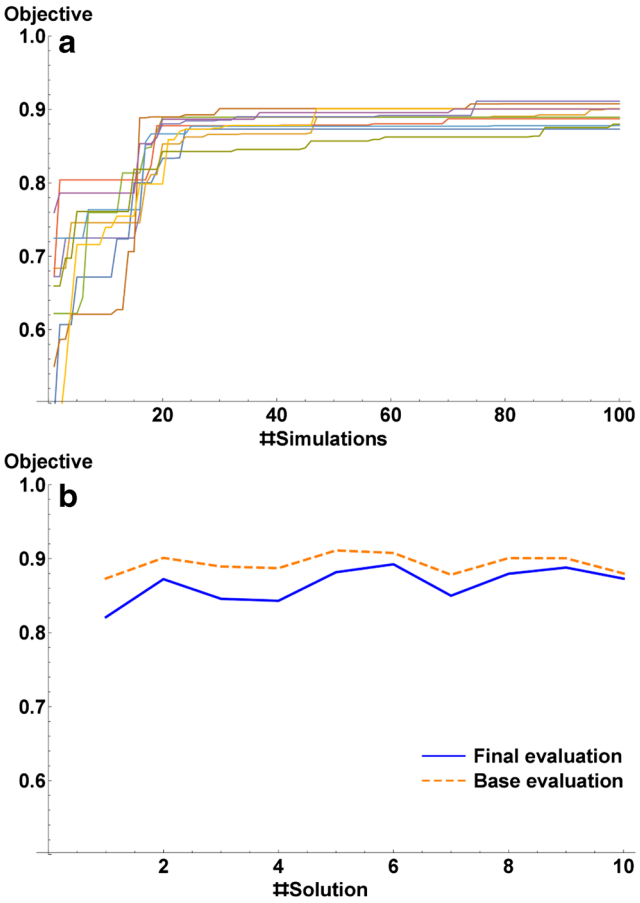


Fig. 16 Optimization of the pork belly pick and place action. **a** Each graph shows the best score achieved as the optimization algorithm progress over the 100 iterations. The 10 different graphs represent the 10 optimization runs. **b** The 10 resulting optimal solutions are evaluated 1000 times to determine the very best with a more extensive evaluation. The dashed orange line shows the scores of the optimized parameters based on 10 evaluations and the blue line shows the scores of the same parameters based on 1000 evaluations

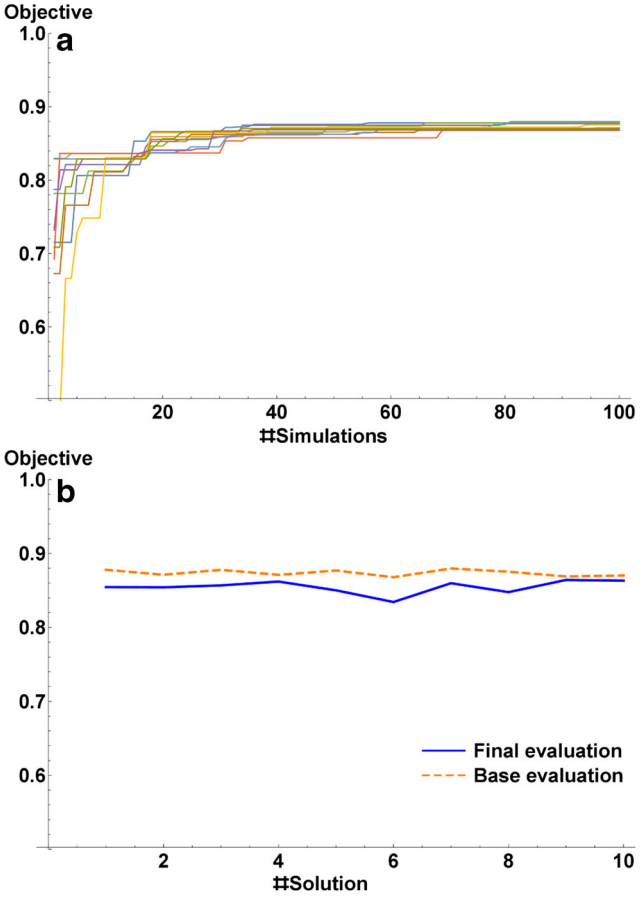


Fig. 17 Optimization of the pork loin pick and place action. The graphs are similar to Fig. 16

1000 evaluations are correlated. However, picking a solution based on 1000 evaluations changes the best pick from solution 5 to solution 6, so the final evaluation improves the choice slightly.

The optimization process for the pork loin case is illustrated in Fig. 17. This case appears easier since the objective scores during optimization converge more quickly. The best solution is at index 9, where the objective score evaluated based on 1000 simulations result in 0.864. Furthermore, the performance of the optimized parameter sets is more similar compared to the pork belly case. Again the best solution changes from solution 7 to 9 when 1000 simulations are used, so again the final evaluation improves the choice slightly.

The optimal parameter sets for the two cases are shown in Table 4.

8 Real World Evaluation

In this section, the real world evaluation of the robot solutions proposed in this paper is discussed. First, we discuss how the grasp strategy was fine-tuned and evaluated on a physical prototype at a Danish slaughterhouse. Then we discuss the evaluation of the placement strategy, which was optimized in simulation. The optimized solutions are evaluated for pork bellies and pork loins.

During the evaluation, we also measured the timing of the different steps. Generating the image pairs with different projected phases took 4 seconds. The algorithm converting the images into a point cloud, segmenting the meat surface and generating a robot trajectory took 4.5 seconds. The robot motion itself varied from 10-11 seconds.

8.1 Grasping Different Pork Cuts

The first part of the experiments was done to determine a reliable grasp strategy for the physical prototype. This was difficult to optimize in simulation due to many subtle effects playing a role in the success of each grasp. To capture

Table 4 Control parameters used during real world trials. Pork belly and Pork loin refer to the optimal parameter sets for the two cases

	Gripper	Rolling grasp				Placement						
	d	d_{ideal}	w	gl	gr	d_{x1}	d_{y1}	d_{z1}	θ	b	d_{x2}	d_{y2}
Pork belly	170mm	22.7mm	0.78	199mm	16.5°	9.6mm	0.3mm	65.9mm	0.2°	1.33	14.4mm	0.3mm
Pork loin	170mm	42.7mm	0.99	102mm	15.3°	.5mm	0.7mm	100.0mm	0.0°	1.28	15.6mm	1.8mm
Default	130mm	0mm	0.5	150mm	20°	0mm	0mm	0mm	0°	0	0mm	0mm

all these effects in simulation would be computationally intractable.

During the real world trials of the grasp strategy, seven different cuts of pork were grasped. These are all shown in Fig. 18, the backs and loins are thicker than the bellies and therefore more rigid. The bellies are wider and thinner and therefore tend to be quite flexible. The heavy bellies are overall larger than the narrow bellies. The undercut bellies tend to be the least rigid since some of the meat structure on the meat side is removed.

During the grasps, several types of failures occurred, to better analyze the solutions we have split the grasp results into 4 categories. These categories are 3 different failure types and success, S. All the categories are illustrated in Fig. 19. The first failure type is failure before the lift, FBL. This refers to failures where the gripper tool is unable to establish vacuum before lifting the meat. The second failure type is failure after the lift, FAL. This refers to the suction cups losing vacuum after the meat is lifted and separated from the piece below. The last failure type is failure due to multiple lifts, FML. This refers to failures where two meat pieces or a meat piece and the box stick together. This can cause the gripper to lift both objects, which is undesirable.

After some initial trials and fine tuning of the rolling grasp strategy, we evaluated it on all 7 cuts of pork. The success rate and the failure causes of the grasps are shown in Fig. 20. For most cuts between 40 and 50 trials were done, but for pork bellies with skin (Fig. 18g) we only did 15 since this was clearly easier than all other cases.

The results show that the undercut pork bellies are significantly more difficult to lift than the single ribbed bellies. This is because they contain less structure and thus are more flexible. During the lift, this extra flexibility makes it more likely that the meat deforms at the suction cups and allow air to flow in.

It can also be seen that it is only undercut bellies that fail due to the gripper lifting multiple objects. This is again due to the lack of meat structure which makes it more likely

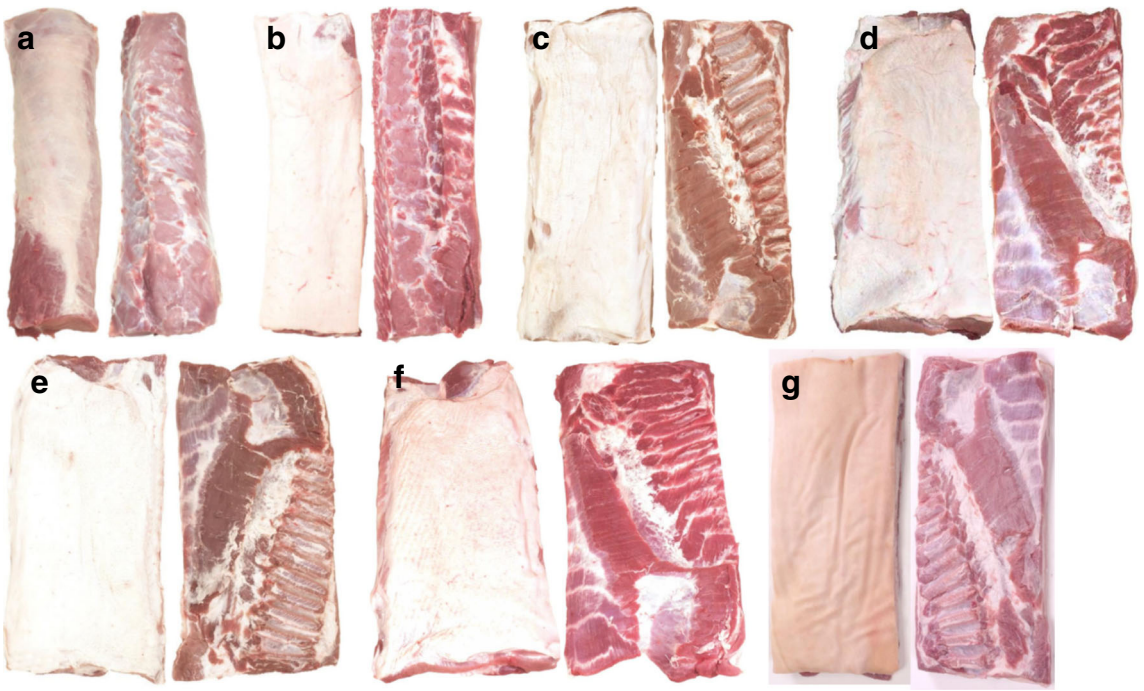


Fig. 18 The pork cuts tested during grasp evaluation. **a** pork loin, **b** pork back, **c** single ribbed narrow belly, **d** undercut narrow belly, **e** single ribbed heavy belly, **f** undercut heavy belly, **g** single ribbed narrow belly with skin

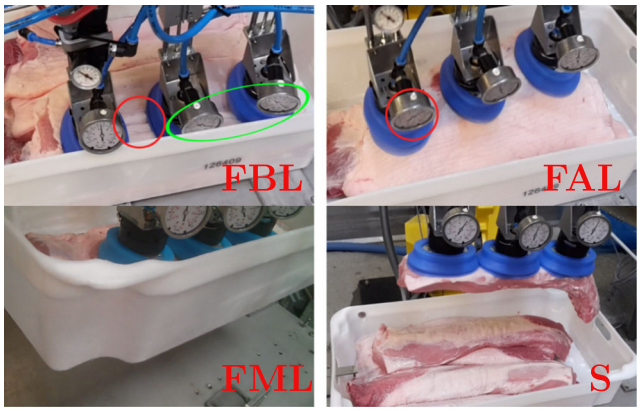


Fig. 19 Failure and success categories. FBL) the green circle highlights the vacuum gauges which show vacuum is never established. The reason for this is the large bulges highlighted by the red circle. FAL) the lift starts well, but vacuum is lost at the vacuum gauge highlighted by the red circle. FML) the box is lifted with the meat. S) a successful lift of the pork loin

that a vacuum is formed between two meat pieces such that they stick together. Besides the single ribbed pork bellies the system also handles pork backs well, and as seen from the failure types pork backs are never dropped during the lift. This is because this cut is thicker and more rigid, thus the meat is less prone to deform and allow air to flow into the suction cups. However, pork backs are also more narrow and thus it is more likely that the suction cups are slightly misplaced before the grasp. The system also handles pork loins well and since they deform even less than the pork backs, the suction cups almost always create and maintain a stable vacuum.

After the grasp strategy was tested, the next step was to optimize the entire pick and place action in simulation. Since pork bellies are the most common cut at the "Danish Crown" slaughterhouse we decided one of the test cases should be a single ribbed heavy belly (Fig. 18e). The reason for picking this particular belly cut is that the vacuum gripper is more likely to work on single ribbed cuts. Furthermore, since it is wider it is more difficult to control during the placement which makes it more interesting from a scientific perspective.

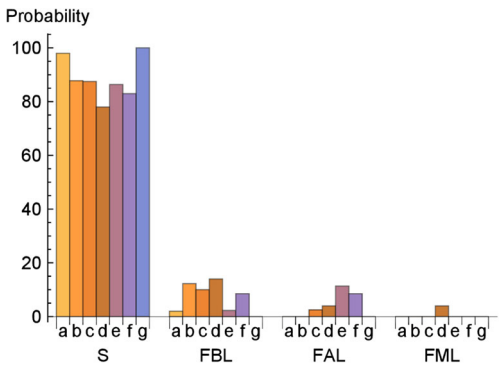


Fig. 20 Success and failure rates of the grasp strategy. **a**, **b**, **c**, **d**, **e**, **f** and **g** refers to the cuts listed in Fig. 18. S is successful grasp, FBL is failure before lift, FAL is failure after lift and FML is failure due to lifting multiple pieces

Besides the pork belly we also picked the pork loins as a test case, the reason for this is that the loins are the cut that differs most from the bellies. Thus this is the best cut for illustrating the versatility of the system.

8.2 Placement Quality

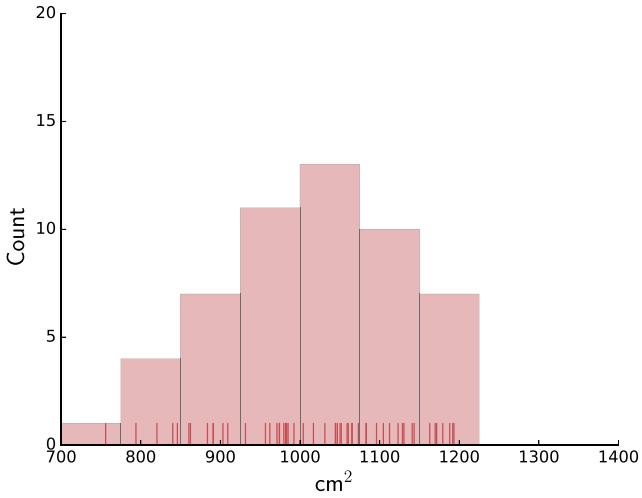
After the rolling grasp strategy was fine-tuned and tested in real world trials (see Section 8.1) the next step was to optimize the parameters relevant for the placement in simulation. This was done as discussed in Section 7. After the optimal parameter set was found, the next step was to evaluate it in the real world and determine whether it leads to better performance. We evaluated the default parameter set and the optimized pork belly parameter set from Table 4 for picking and placing pork bellies in the real world.

We evaluate the quality of each strategy via running a series of trail grasps. We allow the robot to pick the meat and place it as intended on a table. Then we acquire an image of the meat which we can use to quantify the results. The idea is that the ideal pose should be the meat lying completely flat without any folds on the delivery table. This means that the meat's visible surface will be maximised. So by taking an image of the meat in it's delivered pose and quantifying it's surface area, we can quantity the quality of delivery.

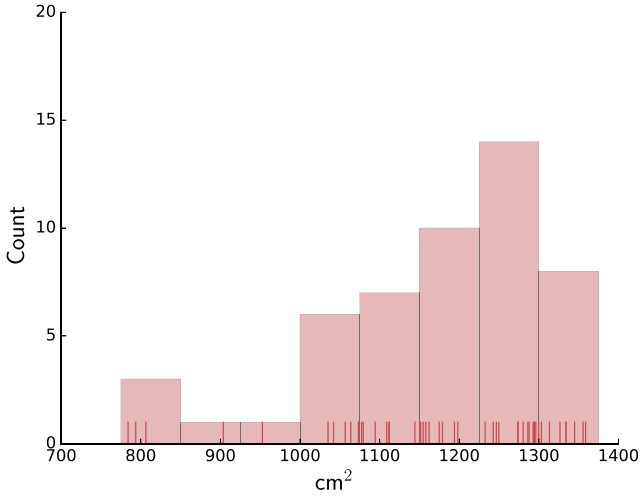
After the robot has transported the meat, we take a photo. We also ensure that a calibration artificat (checkerboard) is present near the meat. By using the artifact we can deduce the meat's physical size from its image space size as well as its physical location. Figure 21 shows an example of one such photography.

We have this experiment for pork belly cutouts and Fig. 22 shows the resulting distribution of the meat's visible surface area after delivery for two strategies: unoptimized and optimized via the previosly described simulation framework. The optimized strategy shows a consistenly higher mean of surface area compared to the unoptimized strategy. The mean being 151cm^2 higher. We can conclude

Fig. 21 Example of the image data collected for our analysis



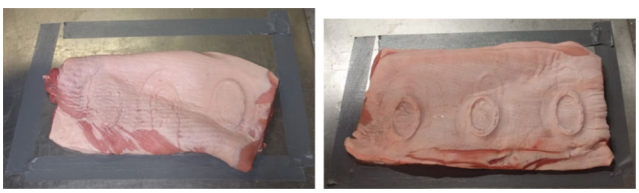
(a) Unoptimized



(b) Optimized

Fig. 22 Distribution of meat area after delivery for the two strategies

that optimized strategy is better at maximising the visible area and thus at placing the meat in an optimal flat pose. Qualitative inspection supports this conclusion as the one side of the pork belly is consistently folded for the unoptimized strategy. Figure 23 shows an example of this



(a) Unoptimized (b) Optimized

Fig. 23 Examples of pork belly pose after delivery. Unoptimized grasping consistently folds the meat, whereas the optimized strategy delivers a consistently flat pose

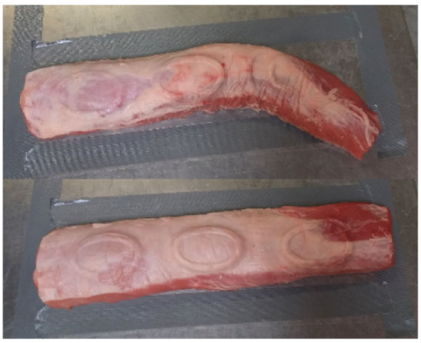


Fig. 24 Top shows a bad placement. Bottom shows a satisfactory pose

along with a succesful example from the optimized strategy. As such we can see a clear improvement in quality by employing parameters obtained from the simulation-based optimization.

As shown by the results in Fig. 20, the system is fairly generic and capable of grasping many variations in cutout shape and size. To illustrate that the placement strategy is also generic, we tested it for the pork loins as well. This was again done by optimizing the placement strategy in simulation and then testing the solution in real world trials. The optimized parameter set is listed as Pork loin in Table 4. Using this parameter set, we achieved a satisfactory placement for approximately 98% of the pick and place operations. A failure and success case is shown in Fig. 24.

9 Conclusion and Future Work

In this work, we have presented a generic solution for doing pick and place operations with meat pieces. Furthermore, we have presented a simulation-based optimization procedure for tuning the generic solution to specific use cases. Finally, the resulting solutions have been evaluated in the real world to validate the approach.

To enable the robot action, the first step was to design a vision system for detecting the meat. The vision system developed for this work is able to generate precise point clouds of the 7 pork cuts tested. Furthermore, a segmentation algorithm was designed, which is able to segment the top surface of all 7 pork cuts.

To move the meat a robot and a suction based gripper tool was used. The robot motion for moving the meat is based on the segmented point cloud from the vision system. Furthermore, it is based on a rolling lift which allows air to flow below the meat piece to avoid it sticking to the surface below. The placement strategy is designed as a simple draping like motion to place the meat piece stretched out on a table.

The entire pick and place action was optimized in simulation to determine the most robust action for placing the meat flat on a table. The resulting solution was tested in the real world. This solution was compared to a non-optimized solution and it is shown that the optimized solution improves the performance by stretching the meat more in the real world as well.

To show the generality of the entire approach, we also optimized it for moving pork loins. For this case, we achieved a success rate of 98% for placing the pork loins nicely on a table.

In future work, we intend to extend the optimization framework to model more grippers and manipulation tools. This would enable the framework to optimize solutions for a much broader range of problems within the food sector. Furthermore, if new gripper tools are able to handle sacks or cloth, it would be possible to evaluate the system in substantially different domains and show the broad applicability of the overall approach.

The vision solution used in this work is already fairly generic. However, for it to work optimally it requires static background lighting, which cannot always be garuanteed. A possible solution to this problem would be the light concentration technique of [14], which vastly increases the SNR of the projected pattern thus making the noise from the background illumination irrelevant. Another feasible solution would be to increase acquisition speed via better hardware synchronization. It should be possible to each speeds of 10-20 point clouds pr. second [38]. Such speed would make most background lighting appear approximately constant.

Even though there are some limitations to the presented system, the system and the individual technologies can still help speed up the design and integration of automation systems for handling meat pieces. This is especially beneficial when automating small batch production, where the design and integration cost is a relatively large part of the overall production cost.

Acknowledgments The financial support from the The Danish Innovation Foundation through the strategic platform "MADE-Platform for Future Production" and from the EU project ReconCell (FP7-ICT-680431) is gratefully acknowledged.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Balaguer, B., Carpin, S.: Combining imitation and reinforcement learning to fold deformable planar objects. In: IROS, pp. 1405–1412. IEEE. http://dblp.uni-trier.de/db/conf/iros/iros2011.html#BalaguerC11 (2011)
2. Berkenkamp, F., Krause, A., Schoellig, A.P.: Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. arXiv:1602.04450 (2016)
3. Bodenhagen, L., Fugl, A.R., Jordt, A., Willatzen, M., Andersen, K.A., Olsen, M.M., Koch, R., Petersen, H.G., Krüger, N.: An adaptable robot vision system performing manipulation actions with flexible objects. IEEE Trans. Autom. Sci. Eng. **11**(3), 749–765 (2014)
4. Buch, J.P., Laursen, J.S., Sørensen, L.C., Ellekilde, L.P., Kraft, D., Schultz, U.P., Petersen, H.G.: Applying simulation and a domain-specific language for an adaptive action library. In: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, pp. 86–97. Springer (2014)
5. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: Computer Vision and Pattern Recognition (CVPR) (2017)
6. Calandra, R., Seyfarth, A., Peters, J., Deisenroth, M.P.: Bayesian optimization for learning gaits under uncertainty. Ann. Math. Artif. Intell. **76**(1-2), 5–23 (2016)
7. Costa, A., Nannicini, G.: Rbfopt: an open-source library for black-box optimization with costly function evaluations. Optimization Online (4538) (2014)
8. Eberly, D.: Thin plate splines. Geometric Tools Inc **2002**, 116 (2002)
9. Eiríksson, E.R., Wilm, J., Pedersen, D.B., Aanæs, H.: Precision and accuracy parameters in structured light 3-d scanning. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 40 (2015)
10. Ellekilde, L.P., Jorgensen, J.A.: Robwork: A flexible toolbox for robotics research and education. In: 2010 41st International Symposium on and 2010 6th German Conference on Robotics of Robotics (ISR), (ROBOTIK), pp. 1–7. VDE (2010)
11. Geng, J.: Structured-light 3d surface imaging: a tutorial. Adv. Opt. Photon. **3**(2), 128–160 (2011)
12. Gonzalez-Jorge, H., Rodríguez-Gonzálvez, P., Martínez-Sánchez, J., González-Aguilera, D., Arias, P., Gesto, M., Díaz-Vilariño, L.: Metrological comparison between kinect i and kinect ii sensors. Measurement **70**, 21–26 (2015)
13. Gupta, M., Nayar, S.K.: Micro phase shifting. Proc. IEEE CVPR, pp. 813–820 (2012)
14. Gupta, M., Yin, Q., Nayar, S.K.: Structured light in sunlight. In: The IEEE International Conference on Computer Vision (ICCV) (2013)
15. Hemker, T., Stelzer, M., von Stryk, O., Sakamoto, H.: Efficient walking speed optimization of a humanoid robot. Int. J. Robot. Res. **28**(2), 303–314 (2009)
16. Jensen, S., Wilm, J., Aanæs, c.H.: An error analysis of structured light scanning of biological tissue, pp. 135–145 Springer (2017)
17. Jørgensen, T.B., Holm, P.H.S., Petersen, H.G., Krüger, N.: Intelligent Robotics and Applications: 8th International Conference, ICIRA 2015, Portsmouth, UK, August 24-27, 2015. Springer International Publishing, Cham (2015)
18. Jørgensen, T.B., Debrabant, K., Krüger, N.: Robust optimization of robotic pick and place operations for deformable objects through simulation. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 3863–3870 (2016)
19. Jørgensen, T.B., Pedersen, M.M., Hansen, N.W., Hansen, B.R., Kruger, N.: A flexible suction based grasp tool and associated grasp strategies for handling meat. International Conference on Mechatronics and Robotics Engineering accepted (2017)
20. Jørgensen, T.B., Wolniakowski, A., Petersen, H.G., Debrabant, K., Kruger, N.: Robust optimization with applications to design of context specific robot solutions. Robotics and Computer Integrated Manufacturing Submitted (2017)
21. Kabsch, W.: A solution for the best rotation to relate two sets of vectors. Acta Crystallogr., Sect. A: Cryst. Phys., Diffr., Theor. Gen. Crystallogr. **32**(5), 922–923 (1976)

22. Koh, Y.J., Kim, C.S.: Primary object segmentation in videos based on region augmentation and reduction. http://openaccess.thecvf.com/content_cvpr_2017/papers/Koh_Primary_Object_Segmentation_CVPR_2017_paper.pdf (2017)
23. Kruse, D., Radke, R.J., Wen, J.T.: Human-robot collaborative handling of highly deformable materials. In: American Control Conference (ACC), 2017, pp. 1511–1516. IEEE (2017)
24. Li, Y., Wang, Y., Case, M., Chang, S.F., Allen, P.K.: Real-time pose estimation of deformable objects using a volumetric approach. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), pp. 1046–1052. IEEE (2014)
25. Long, P., Khalil, W., Martinet, P.: Robotic deformable object cutting: from simulation to experimental validation (2014)
26. Loshchilov, I., Schoenauer, M., Sebag, M.: Adaptive coordinate descent (2011)
27. Mesit, J., Guha, R., Chaudhry, S.: 3d soft body simulation using mass-spring system with internal pressure force and simplified implicit integration. J. Comput. **2**(8), 34–43 (2007)
28. Misimi, E., Øye, E.R., Eilertsen, A., Mathiassen, J.R., AAsebø, O.B., Gjerstad, T., Buljo, J., Skotheim, Ø.: Gribbot–robotic 3d vision-guided harvesting of chicken fillets. Comput. Electron. Agric. **121**, 84–100 (2016)
29. Nabil, E., Belhassen-Chedli, B., Grigore, G.: Soft material modeling for robotic task formulation and control in the muscle separation process. Robot. Comput. Integr. Manuf. **32**, 37–53 (2015)
30. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 davis challenge on video object segmentation. arXiv:1704.00675 (2017)
31. Posdamer, J., Altschuler, M.: Surface measurement by space-encoded projected beam systems. Comput. Graphics Image Process. **18**, 1–17 (1982). https://doi.org/10.1016/0146-664X(82)90096-X
32. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China (2011)
33. Rusu, R.B., Cousins, S.: Region growing segmentation. http://pointclouds.org/documentation/tutorials/region_growing_segmentation.php (2017)
34. Schulman, J., Lee, A., Ho, J., Abbeel, P.: Tracking deformable objects with point clouds. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 1130–1137. IEEE (2013)
35. Tappen, M.F., Freeman, W.T.: Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In: Null, p. 900. IEEE (2003)
36. Tesch, M., Schneider, J., Choset, H.: Using response surfaces and expected improvement to optimize snake robot gait parameters. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1069–1074. IEEE (2011)
37. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. In: BMVC (2017)
38. Wilm, J., Olesen, O.V., Larsen, R.: Slstudio: open-source framework for real-time structured light. Proceedings of the 4th International Conference on Image Processing Theory, Tools and Application (ipta 2014) p. 7002001. https://doi.org/10.1109/IPTA.2014.7002001 (2014)
39. Wolniakowski, A., Jorgensen, J.A., Miatliuk, K., Petersen, H.G., Kruger, N.: Task and context sensitive optimization of gripper design using dynamic grasp simulation. In: 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 29–34. IEEE (2015)
40. Zoumponos, G.T., Aspragathos, N.A.: A fuzzy strategy for the robotic folding of fabrics with machine vision feedback. Industrial Robot: An International Journal **37**(3), 302–308 (2010)

Troels Bo Jørgensen is a PhD student at the Mærsk McKinney Møller Institute, University of Southern Denmark. His research interest include applied robotics, mainly in the fields of dynamic simulation, optimization and workcell design.

Sebastian Hoppe Nesgaard Jensen is a PhD student at the Department of Applied Mathematics and Computer Science, Technical University of Denmark. His main field of research is applied 3D vision which specifically includes structured light scanning, structure from motion and segmentation.

Henrik Aanæs is associate professor in computer vision at the Technical University of Denmark, where he, among others, heads an effort concerned with the industrial application of 3D computer vision, e.g. for robotics and metrology.

Niels Worsøe Hansen Senior Project Manager, Slaughterhouse Technologies, Danish Technological Institute. BSc Mechanical Engineering from Copenhagen University College of Engineering. Main activities and responsibilities is project management, development, and implementation of new technologies.

Norbert Krüger is a professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. He holds a M.Sc. degree from the Ruhr-Universität Bochum, Germany and his Ph.D. degree from the University of Bielefeld, Germany. His research covers computer vision, cognitive systems and applied robotics.