



Towards footwear manufacturing 4.0: shoe sole robotic grasping in assembling operations

Guillermo Oliver¹ · Pablo Gil¹ · Jose F. Gomez² · Fernando Torres¹

Received: 7 September 2020 / Accepted: 24 January 2021 / Published online: 23 March 2021
© The Author(s) 2021

Abstract

In this paper, we present a robotic workcell for task automation in footwear manufacturing such as sole digitization, glue dispensing, and sole manipulation from different places within the factory plant. We aim to make progress towards shoe industry 4.0. To achieve it, we have implemented a novel sole grasping method, compatible with soles of different shapes, sizes, and materials, by exploiting the particular characteristics of these objects. Our proposal is able to work well with low density point clouds from a single RGBD camera and also with dense point clouds obtained from a laser scanner digitizer. The method computes antipodal grasping points from visual data in both cases and it does not require a previous recognition of sole. It relies on sole contour extraction using concave hulls and measuring the curvature on contour areas. Our method was tested both in a simulated environment and in real conditions of manufacturing at INESCOP facilities, processing 20 soles with different sizes and characteristics. Grasps were performed in two different configurations, obtaining an average score of 97.5% of successful real grasps for soles without heel made with materials of low or medium flexibility. In both cases, the grasping method was tested without carrying out tactile control throughout the task.

Keywords Robotics · Grasping · Manufacturing automation · Footwear · Shoe soles

1 Introduction

Traditionally, use of technology in footwear manufacturing has been mainly focused on applying CAD/CAM processes to improve shoe appearance and ergonomics [1]. Technology enables the shoe industry to reduce defects, achieve high-quality standards [2], and perform personalized pieces [3]. Besides, virtual reality (VR) technologies are being

successfully applied to create new footwear models, providing interactive experiences for the visualization of fashion shoes [4].

Throughout the last decades, many efforts have been made by the shoe industry to fully or partially automate some parts of the manufacturing process [5]. However, shoe making involves many tasks, and many of them still employ human operators. Some of these tasks are done by a worker using manual shoe making tools, while others involve human interaction with a specific machine, i.e., material cutting of each shoe pattern, sewing machines for stitching, cementing and pressing machines for assembling parts together, etc. For this reason, the future of automation in footwear industry will require promoting the use of robots [6]. In state of the art, there are innovative solutions combining image processing [1, 7] or 3D visual perception with robots [8], e.g., to generate trajectories to cut leather [9] or for robot shoe-groove tracking from feature extraction on the surface of a scanned shoe upper [10]. Furthermore, we can find shoe glue application systems based on visual techniques using shape reconstruction and guided by robotic end-effectors [11, 12], and kinematic control of robots to perform buffing and roughing operations on shoe uppers [13, 14]. Moreover, robots have also been introduced in

✉ Guillermo Oliver
guillermo.oliver@ua.es

Pablo Gil
pablo.gil@ua.es

Jose F. Gomez
jfgomez@inescop.es

Fernando Torres
fernando.torres@ua.es

¹ Automatics, Robotics and Artificial Vision Lab (AUROVA), Computer Science Research Institute, University of Alicante, San Vicente del Raspeig, Spain

² Footwear Technological Institute (INESCOP), Elda, Alicante, Spain

other finishing operations such as polishing cream and spraying [15] or collaborating with humans in shoe-packaging process [16]. However, there are tasks where robots have not been introduced yet or they have not been correctly adapted to the variability of the manufacturing process.

In this work, we propose a robotic workcell with a novel visual perception method to compute grasping points in two different scenarios that occur in the manufacturing process. The first scenario consists of soles travelling on a conveyor belt, while the second has soles resting on a custom location. Fashion trends induce high variability in shoe shape and make the grasping task challenging. Recently, reinforcement learning techniques driven by computer vision were used to perform isolated operations of shoe grasping [17]. Even with these techniques, it is still difficult to accomplish a re-configurable robotic workcell at a competitive working speed. Our work is inspired by [18], but, unlike this one, our cell works without the support of other specific machines (e.g., for roughing and cementing). In that work, authors tested the proposal with only one type of shoe, while our system has been tested with 20 different models. Footwear grasping is also present in [19], where authors defined a system to perform pick-and-place operations on the last taking 3D model templates as references. We propose to increase system flexibility without using templates or prior knowledge of the shoe model, and without relying on reinforcement learning. Our goal is to move towards a flexible and automated footwear industry 4.0 according to the fundamental underlying design principle presented in [20].

RGBD cameras and robots were successfully used for glue application on shoe soles as shown in [8]. Our proposal allows a robotic arm endowed with a gripper and a RGBD camera to grasp any type of sole regardless of size and model. Usually, size depends on foot length and model defines the kind of material (rubber, plastic, leather, etc.), texture (rough, smooth, etc.), and color. Our method grasps the sole from its contour, avoiding parts with adhesive/glue, so that the gripper is not damaged and the pressing process can be done successfully later. We tested our workcell with 20 real soles of various sizes and characteristics, performing 10 tests per sole in a real environment at INESCOP facilities.

This paper is organized as follows. Section 2 presents the footwear manufacturing process and the current level of automation. Section 3 presents our robotic cell for sole manipulation in assembling operations of footwear manufacturing. Section 4 details the algorithm used in the cell to select the best grasp on soles. Section 5 shows the obtained results using real soles with different features. Finally, Section 6 summarizes our findings and contributions, shows the main limitations of this work, and draws some future research lines.

2 Traditional issues of footwear manufacturing

Shoe making has always been considered a traditional handicraft process. Shoe manufacturing process consists of many operations which depend on the variety of materials (i.e., fabric, plastic, leather, rubber, foams, etc.) the shoe is made of, as well as on its design. Design is very different according to the kind of shoe (dress, sport, boot, sandal, etc.), its application (i.e., business shoes, derby, sneakers, etc.), and fashion trends. Some of the main operations are cutting, stitching, assembling, cementing, and finishing (Fig. 1). Each of them requires several tasks.

Cutting deals with the upper of the shoe and it is composed of tasks such as leather selection, marking the pattern within the selected leather piece, and cutting the shape of the upper. The upper patterns must be cut distributing them along the leather sheet, in order to make the best use of space, but, at the same time, avoiding any flaws in the raw material.

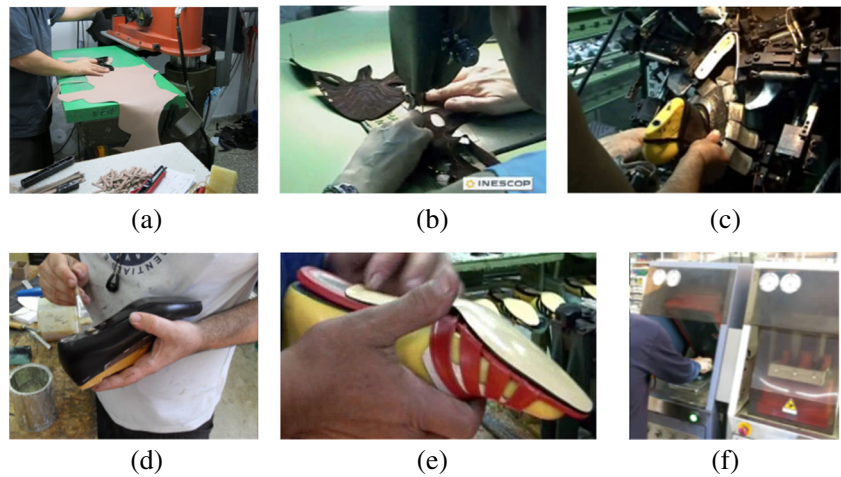
Stitching encompasses all the tasks performed to assemble the pieces that make up the shoe. This operation requires a minimum of two workers, one for part assembling and other for sewing. Furthermore, in this phase, some pieces are split so that the overlap among them does not increase its thickness. Besides, some degree of edge thickness is removed to enable bending motion. Prior to stitching, other tasks, such as stamping of some pieces or punching to add the eyelets where the shoelaces would be, can be required.

During the assembling process, the finished upper is molded into a foot shape using a piece called the last (Fig. 2). To do this, the insole is fixed to the last. Both the upper and insole are fixed to each other with adhesive or nails. Then, a stiffener and a side lining piece are included to give support to the sides of the upper when the last is removed. Finally, exterior and interior pieces are assembled and closed together.

The next phase is cementing. First, the molded upper is bevelled, staples or nails removed from the insole, and the surface smoothed. Afterwards, adhesive is applied on previously defined areas of the outsole to fix it with the assembled insole and upper. Sometimes, adhesive activation with heat is required before proceeding with the binding together of the insole and outsole. Last, the outsole is placed on the insole and pressed to consolidate their anchoring. Finally, the finishing process comprises adding finishing touches such as attaching heel, extracting the shoe from the last, ironing, and polishing.

In the last years, footwear manufacturing has become a hybrid process combining three kinds of operations: handmade, machine-aided, and completely automatic with appropriate specific machines. Table 1 summarizes the main

Fig. 1 Several shoe manufacturing stages. **a** Cutting. **b** Stitching. **c** Shaping. **d** Adhesive application. **e** Sole assembling. **f** Pressing



characteristics of the stages shown in Fig. 1. Nowadays, cutting operations have been fully automated with shoe pattern design software and laser cutting robots, while stitching is mostly made by humans with little or almost no machine assistance. On the other side, assembling and cementing are often performed by humans using non-automatic machines.

This work focuses on presenting an automation solution based on strategies of robotic manipulation and 3D computer vision techniques to automate some tasks of the cementing operation. Our aim is to replace operators and machines working at this operation with a combination of robotic arms and fully autonomous machines. The most challenging task of the cementing phase is outsole manipulation. Outsoles are very thin objects, which makes them difficult to grasp.

3 Fabrication cell architecture

3.1 Virtual scenario

We assembled a virtual scenario of the fabrication cell (Fig. 3) to assess its performance. The virtual scenario emulates the real setup of the cell and it will be imported

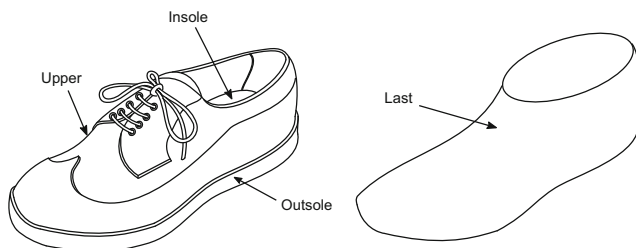


Fig. 2 Diagram of a shoe and its last

into Robot Operating System (ROS) [21] to allow simulated runs of the process and diagnose potential issues before performing real experiments.

Our fabrication cell is made up of several stations with well-differentiated tasks: scanning and grasping station, rotation station, and assembling station. Each of them uses different visual perception systems to accomplish the interaction with the environment and in each one the number of robots involved is different. First station performs three main tasks on the outsole: scanning, applying glue, and picking up. We use two robots and a laser digitizer for this task. Once the sole is picked up, it is transferred to the rotation station. This station consists of a specially crafted place easily accessible from above and below. This allows the sole to be picked up a second time from below to be correctly orientated to carry out the assembly. In the assembly station, the sole is left on a last to be pressed in a future extension of the cell. For these last tasks, we used a single robot with eye-to-hand configuration and a RGBD camera instead of the laser digitizer.

3.2 Real implementation

First, when a sole is placed on the starting position, the conveyor belt turns on and the process begins. For outsole scanning, we use a Gocator 2350D 3D Laser Line Profile Scanner containing a megapixel image sensor that provides up to 1280 points per profile with a resolution of 0.150 mm along the line and 0.019 mm in depth. The scanner is mounted on a structure and attached to a certain position on the conveyor belt. Once the outsole comes out from the scanning box, the belt stops and a Comau Smart Six-6-1.4 robot with a nozzle on its end-effector applies hot glue to it. Finally, the outsole reaches the pickup zone and an UR10 CB3 robot with a Robotiq 2F-140 gripper picks it

Table 1 Characterization of shoe manufacturing stages according to the required skills and the automation level more frequently used

Stage	Skill	Precision	Force	Machine-aided
Cutting	Low	Medium	Low	Laser cutting machine/robot
Stitching	High	High	Low	Sewing machine
Assembling upper and insole	High	Medium	Medium	Stapler/pre-forming/pulling over machine
Applying adhesive (cementing)	Medium	Medium	Low	Manual with hand tools
Sole assembling (cementing)	High	High	Low	Manual without hand tools
Pressing (cementing)	Low	Medium	Low	Pressing machine
Finishing	High	Medium	Low	Iron/cutter/spray gun/brush machine

up. In this first station, we will focus on the grasping of the outsole from the conveyor by the UR10 robot. Our input is the digitized outsole obtained from the scanner and the conveyor position obtained from the encoder of its electric motor.

We can make a Gantt chart of the activities performed in this fabrication cell to show their dependencies and what we will focus on (Fig. 4). Once the outsole is scanned, the digitized data is available and the grasping point process calculation starts. Grasping points must be calculated during the time required for the application of hot glue and before the outsole reaches the pickup zone, in order to avoid idle times and perform the process as fast as possible. This requirement makes the grasping point calculation our critical task. It is one of the main contributions of this work. We must find a way to calculate these grasping points in a fast and effective manner from the point cloud acquired from the digitizer.

Once the sole is grasped and picked up, it is directly transferred to the rotation station. The sole is now resting

on two metal rods that allow it to be easily accessed from below. In order to be assembled on the last, the sole must be picked up from below. The UR10 CB3 robot has a Intel RealSense D415 depth camera attached to its end-effector and it is positioned below the sole to take a picture of the inferior part of it. Grasping points are now calculated again to pick the sole from this new position and transfer it to the assembly station. The process ends with the sole laying on the last to be pressed in a future stage.

3.3 ROS implementation

We will reproduce the grasping task of this scenario in ROS, with the aim of simulating, and eventually controlling, the real robot during the process (Fig. 5).

For this task, we use Ubuntu 16.04 and ROS Kinetic. We use RViz, the primary visualizer in ROS, with the *MoveIt! Motion Planning Framework* [22] plugin to build a virtual environment that will be ultimately configured to connect to the physical robot and perform the real motion. We import all the CAD models of the physical bodies interacting in the scenario to the simulation. While the UR10 robotic arm, the 2F140 gripper and the RealSense D415 camera have ROS models available from their manufacturers, we must also import several custom bodies into ROS: conveyor belt, scanner box, UR10 structure, rotation station, and assembly station.

3.4 Calibration

For the real setup to work correctly, the relative positions of all the elements must be correctly defined (Fig. 6). In order to make the assembly easily reproducible in case of an eventual relocation, all of the elements are fixed to a custom structure that lays underneath, which makes calibration easier avoiding undesired relative movements of the elements.

First of all, we define the global reference frame, F_o , at the base of the UR robotic arm. The transformation e_oT defines the position and orientation of the reference frame end-effector of the robotic arm, F_e , and it is calculated using

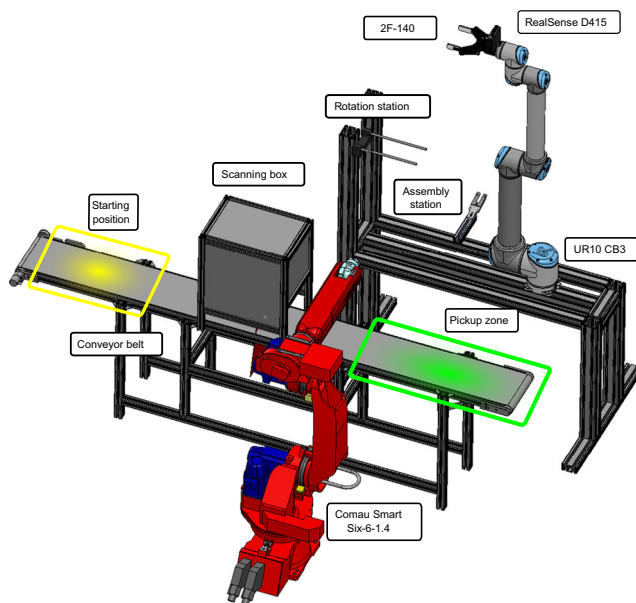
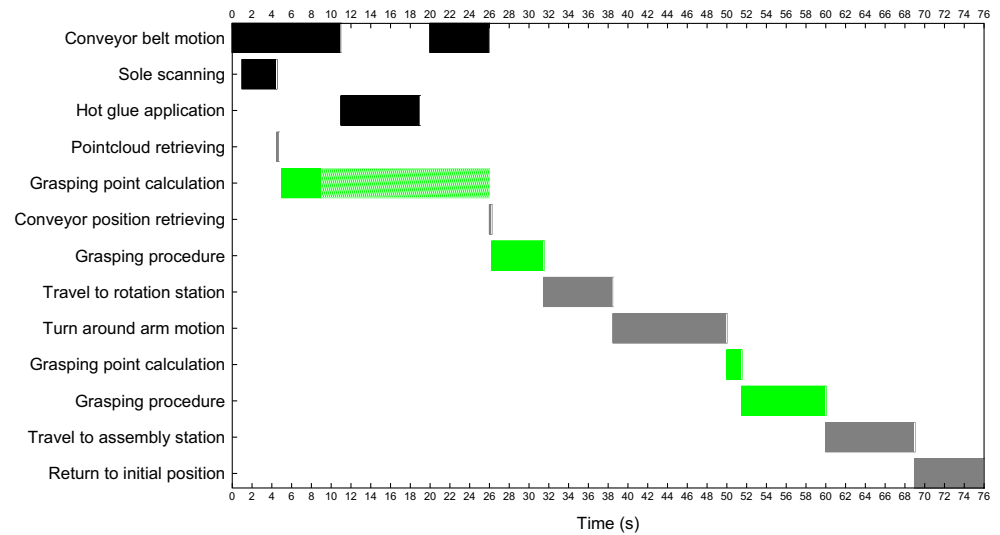
**Fig. 3** Virtual scenario

Fig. 4 Gantt chart of the fabrication cell. Color code: *green*, activities developed in this work; *striped green*, critical task maximum duration; *grey*, inputs for developed activities; *black*, other workcell activities



the Denavit-Hartenberg algorithm [23]. The model is fine-tuned using the calibration parameters retrieved from the robot's controller. We have calibrated the position of the laser scanner reference frame, F_s , using the CAD model of the virtual scenario and verified it using the robot's end-effector, therefore obtaining the transformation matrix s_oT . Glue application, which is not in the scope of this work,

requires the transformation s_rT from the COMAU reference frame, F_r , to the scanner reference frame, as well as the position of the COMAU end-effector frame F_n , which are calculated the same way as before.

The final step, and probably the most important one, is camera calibration. We are obtaining depth pictures from the camera with a height of 720 pixels and a width of

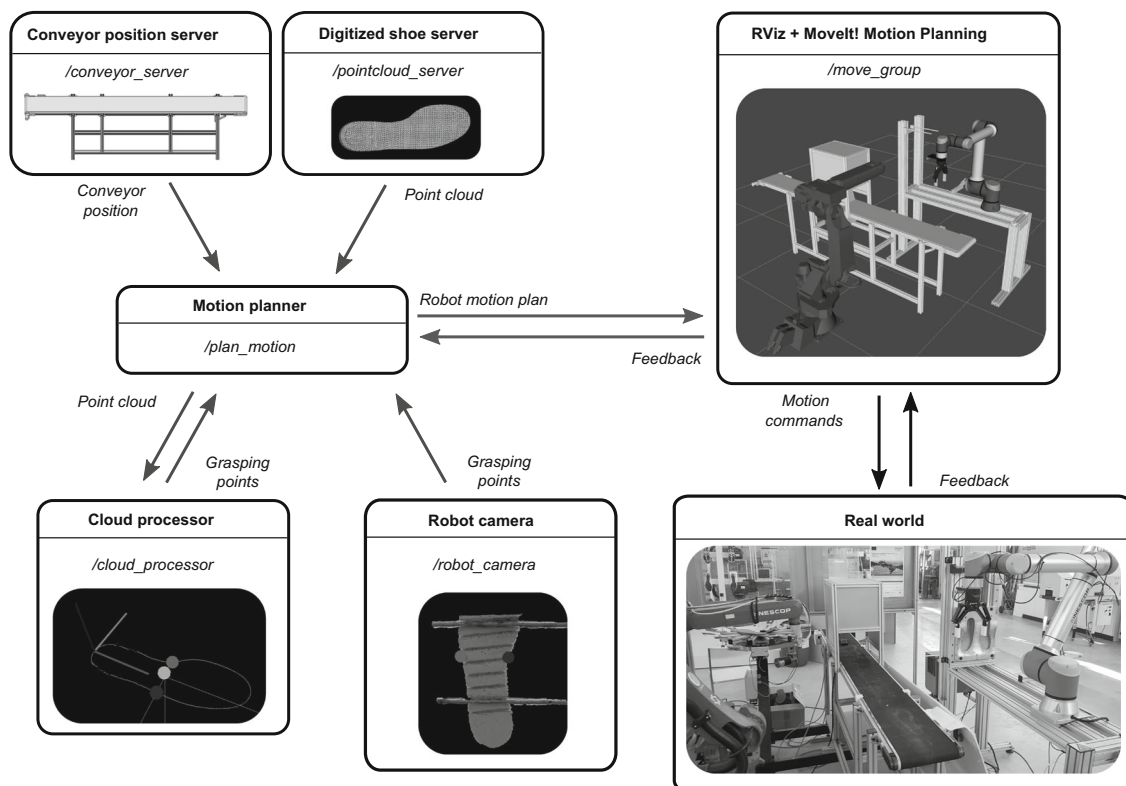


Fig. 5 Layout of the ROS implementation

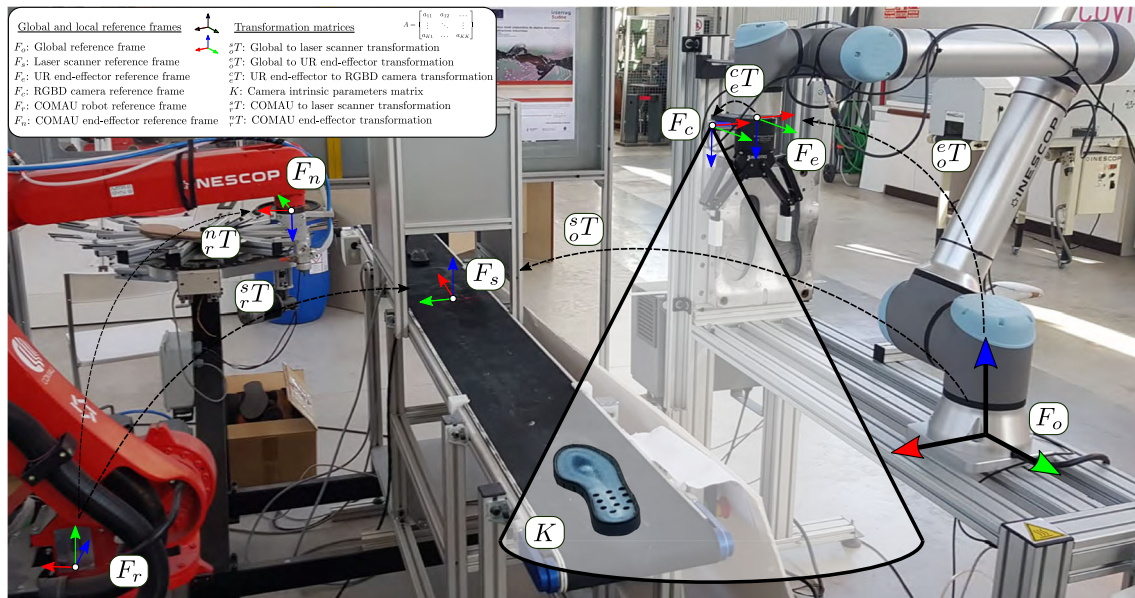


Fig. 6 Real scenario calibration scene

1280 pixels. The camera intrinsic parameter matrix, $K = \begin{pmatrix} 916.46 & 0 & 633.48 \\ 0 & 914.11 & 367.75 \\ 0 & 0 & 1 \end{pmatrix}$, is obtained from the camera's firmware, along with depth-scale parameter ($d = 0.001$). Using this information, depth images are transformed to point clouds by our software. We have used a custom software to obtain the camera's extrinsic parameters. We need to obtain the transformation c_eT from the camera reference frame, F_c , to the UR end-effector frame. This problem is usually referred as camera hand-eye calibration with a robot-mounted sensor (eye-in-hand). In order to do this, a calibration checkerboard grid is located on a known position on the conveyor belt structure and the robot end-effector is moved to several positions where this board is visible, where we take pictures at each of those positions and detect the center of the board in every picture. Using this information, we can fit a transformation between the end-effector position and the camera origin, obtaining the extrinsic parameters of the current camera configuration. A point expressed in the camera's reference frame, p_c , is converted to global coordinates, p_o , using: $p_o = {}^e_oT {}^c_eT p_c$.

4 Robotic grasping algorithm

4.1 Grasping points

In the scanning and grasping station, we use a custom version of the *GeoGrasp* software modified for the specific task in hand. *GeoGrasp* [24] is an algorithm designed for

the computation of grasping points for unknown objects using a point cloud acquired from a single partial view of a scene. Even though originally designed for usage with RGBD cameras, it has been adapted to process point clouds obtained from a laser scanner. It is an important contribution of this work.

This algorithm extracts geometric features, i.e., centroid and first eigenvector, from the object point cloud using principal component analysis (PCA). We use standard methods from Point Cloud Library (PCL) [25] to perform PCA on the point cloud. Using these features, the algorithm obtains two candidate regions Q_1 and Q_2 . These regions are determined by the intersection of the outermost part of the cloud with a plane defined by the centroid c and the first eigenvector v , considering gripper grasping tip width. The algorithm ranks combinations of points from these regions based on local curvature using a custom ranking function. The best-ranked pair guarantees the most stable grasp configuration $G = \{g_1, g_2\}$, given the view conditions (Fig. 7). A more detailed explanation can be found at [24].

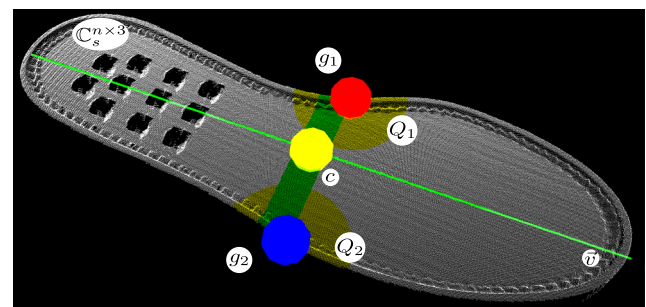


Fig. 7 GeoGrasp algorithm on a shoe sole

Unlike point clouds obtained from RGBD cameras containing a few thousand points, those obtained from laser line profile scanners have a much bigger resolution, usually containing several hundred thousand points for similar dimension objects. We have tested these extremely dense point clouds with this algorithm and it is unable to process them. We could only achieve grasping point generation with aggressive voxelization, reducing significantly point cloud size and obtaining grasping points inside the contour of the outsole, making them unsuitable for this application.

Assuming the object to grasp has a uniform density and no significant differences in height, which is normally the case for an outsole, we can keep only the outermost points of the object. This contour is exactly the place where the outsole should be grasped. Under these assumptions, the center of gravity of the contour is an acceptable approximation of the center of gravity of the original object. We will obtain the contour of the point cloud taken as a 2D hull on the dominant plane, reducing point cloud complexity and significantly decreasing processing time. The algorithm performs PCA feature extraction, i.e., centroid and first eigenvector, from the contour to calculate grasping points. However, if the sole has very distinct characteristics, deviations introduced by hull generation are corrected with those extracted from the original cloud (Algorithm 1), using these features to calculate the grasping points on the outsole contour. Figure 8 shows that this method is useful to correct deviations produced by gaps in hull generation (Fig. 8d), non-uniform hull sampling (Fig. 8b), or by non-standard sole typologies (Fig. 8c).

In order to extract the object's contour, we must obtain a hull of it. To achieve this task, we will use alpha shapes,

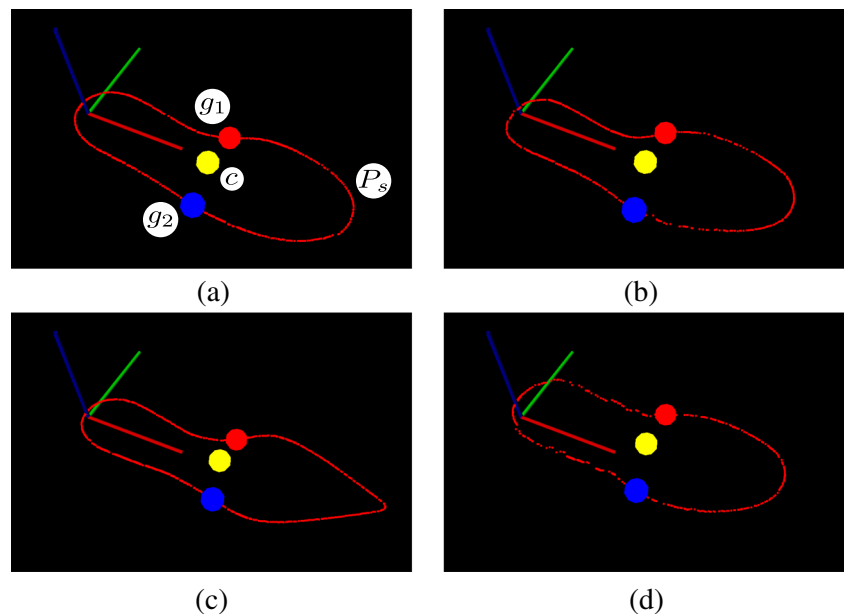
a well-known concept in computational geometry introduced in [26], and for which there is a straightforward implementation in PCL. First, we will define α as the curvature of a disk of radius $r = 1/\alpha$. This is the main parameter of the alpha shape. Taking as input a finite set of points, two members of it make up an edge of the alpha shape if a disk of radius r with those two points lying on its boundary, and containing none of the other points of the set, exists. Alpha value determines two types of hulls: convex and concave hulls.

The least computationally expensive process is convex hull generation. This corresponds with a value of $\alpha = 0$, causing the disk to become a half-space. Intuitively, we can think of a convex hull as the smallest polygon enclosing a given distribution of points. This hull is greater than (or equal to) the real contour; some points that belong to the real contour are left inside, but none outside of its borders. When the contour has a certain concavity, a convex hull is not a valid approximation to the real contour. We will perform this using the *ConvexHull* function implemented in PCL, which uses the *QuickHull* method [27]. This algorithm reduces point cloud size in a short amount of time.

Concave hull algorithms are significantly more complex and require much more computational power. In this case, PCL also provides a function to calculate concave hulls, *ConcaveHull*, from the same authors [27]. This case corresponds with an alpha value of $\alpha > 0$, and we must provide this value to the function. An optimum value for sole hull generation of $\alpha = 0.005$ was obtained in a previous experimentation found in [28].

Once this concave hull is obtained, it is used as input for the *GeoGrasp* method as shown in Algorithm 1. In our

Fig. 8 Example of generated grasping points G obtained using Algorithm 1 on concave hulls with PCA correction. **a** BallerinasEVA outsole. **b** Unisa outsole. **c** 9100 outsole. **d** Trip outsole



proposal, the PCA values of the original cloud $\mathbb{C}_s^{n \times 3}$ are used as input to correct those of the concave hull \mathbb{P}_s and grasping points G are calculated (Algorithm 1).

Algorithm 1 Conveyor grasping points generation.

Require: Generate grasping points for conveyor station

Input: $\mathbb{C}_s^{n \times 3}$:= Object point cloud from laser scanner

Output: $G = \{g_1, g_2\}$:= Conveyor grasping points

- 1: **Calculate** $PCA(\mathbb{C}_s^{n \times 3})$
 - 2: **Compute** α -shape polygon $\mathbb{P}_s(\mathbb{C}_s^{n \times 3})$
 - 3: **Initialize** *GeoGrasp* with \mathbb{P}_s
 - 4: **Correct** \mathbb{P}_s PCA values with $PCA(\mathbb{C}_s^{n \times 3})$
 - 5: **Calculate** G using *GeoGrasp*
 - 6: **Return** G
-

For the rotation station, we use *GeoGrasp* on the point cloud $\mathbb{C}_c^{n \times 4}$ obtained from the RGBD camera mounted on the robotic arm's end-effector. In this case, the obtained point cloud has such a size that it can be processed directly by the algorithm. We must trim the raw point cloud to remove the background and select only the region of interest where the object appears for the algorithm to calculate the grasp in that region. For this purpose, we use geometric segmentation to select the range in three-dimensional coordinates where the grasping calculation should focus on. Thus, we compute $\mathbb{D}^{k \times 4}$ which represents the region of interest. In this case, it is the place between the metal rods shown in Fig. 3, which is where the gripper will access the lower part of the sole to perform the grasp. Workflow is shown in Algorithm 2.

Algorithm 2 Rotation grasping points generation.

Require: Generate grasping points for rotation station

Input: $\mathbb{C}_c^{n \times 4}$:= Object point cloud from RGBD camera

Output: $G' = \{g'_1, g'_2\}$:= Rotation grasping points

- 1: **Perform** geometric segmentation $\mathbb{D}^{k \times 4} \subset \mathbb{C}_c^{n \times 4}$
 - 2: **Initialize** *GeoGrasp* with $\mathbb{D}^{k \times 4}$
 - 3: **Calculate** G' using *GeoGrasp*
 - 4: **Return** G'
-

4.2 Kinematic analysis and planning

Shoe soles have a wide variety of sizes and shapes, we must select a gripper capable of performing grasps with different widths and depths. Our dataset is composed of sole samples with a height ranging from 0.5 to 3.5 cm and a width between 5 and 8.5 cm at their central section.

We have selected the 2F-140 gripper from Robotiq which has an maximum opening range of 14 cm and an available CAD model from the manufacturer. We can easily determine the kinematics of the gripper to obtain analytical

expressions for the opening of the gripper (Eq. 1) and the distance from the base to the tips of the fingers (Eq. 2).

$$\text{Opening} = 2(L_{2x} + L_2 \cos(\Theta_2 + \Theta_o) - L_p) \quad (1)$$

$$\text{Height} = L_{2y} + L_2 \sin(\Theta_2 + \Theta_o) + L_h \quad (2)$$

The gripper is symmetric left to right (Fig. 9). For this analysis, we reduce the gripper mechanism to a 2D planar four-bar linkage on the right symmetric part. It is a quadrilateral mechanism with a parallelogram configuration. The lengths of its sides are equal in pairs; this causes the mechanism to have purely rotational motion around the anchor points while the orientation of its end-effector is fixed. The outermost bar has a passive degree of freedom that allows the gripper to adapt to the surface of the object it is grasping. We must also consider that the angle expected by the gripper is relative to the maximum opening angle.

This gripper has six parameters and one degree of freedom. The first two parameters, L_{2x} and L_{2y} , are the coordinates of the pivoting point of the mechanism in the local coordinate reference frame. L_2 parameter defines the finger length, while L_p and L_h are the fingertip pad width and length respectively. The maximum opening is 14 cm, and corresponding with $\Theta_2 = 0$. Θ_o is the angular offset of the finger with respect to the local x-axis at the starting position. From the desired grasp width, we calculate the required distance of the robot end-effector to the sole to perform a successful grasp.

Execution of grasps on the real implementation has shown to be dependent on the gripper characteristics, beyond the correctness of the calculation of the grasping points. The 2F-140 gripper has rigid thumbs that are unable to adapt to the shape and materials of the soles, producing slippage when performing the grasp and pickup of soles from the conveyor. We developed a specially crafted thimble printed using a flexible plastic material to increase the friction and obtain a better fit to the sole shapes. All the grasps in the conveyor and the latter rotation station were performed with a thimble-equipped gripper, calibrating gripper fingertip parameters L_p and L_h to account for it.

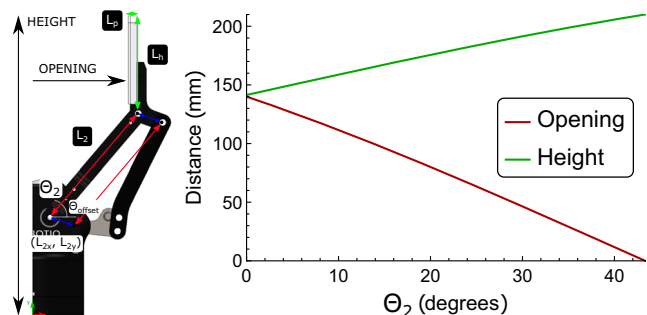


Fig. 9 Kinematic characteristics of the gripper

To assure repeatability in an industrial environment, nothing can be left to chance or sheer luck. The *MoveIt! Motion Planner* uses the Open Motion Planning Library (OMPL) based on randomized motion planners. It takes starting and target points (or starting and target robot poses) as input and outputs a suitable trajectory: a series of intermediate positions (or robot poses) the robot has to reach in order to perform the motion. In most cases, when planning the motion of the robot from one location in Cartesian space to another, it will find a smooth trajectory following the shortest or optimal path (resembling a straight line if there are no collision obstacles). However, due to the random nature of the planner, it can find a path that performs some strange or sharp motions outside of the predictable trajectory. Despite that the working range can be limited by safety plane boundaries, these uncontrolled motions can be dangerous in an industrial environment and when working in a location frequented by human operators.

Algorithm 3 Planned motion checker.

Require: Check suitability of planned motion
Input: $\mathbf{q} = (q_1, q_2, q_3, q_4, q_5, q_6) :=$ Initial robot state
Input: $\mathbf{p} = (x_t, y_t, z_t) :=$ Target point
Input: $T :=$ Planned trajectory
Input: $m_d :=$ Maximum admissible deviation
Output: $s :=$ Suitability of planned motion

- 1: **Calculate** $\mathbf{x} = FK(\mathbf{q}) :=$ Starting end-effector position
- 2: **Calculate** $d_0 = \|\mathbf{p} - \mathbf{x}\| :=$ Shortest distance to target
- 3: **Define** $k = size(T) :=$ Points in planned trajectory
- 4: **Initialize** $d_T = 0 :=$ Trajectory distance travelled
- 5: **while** $i \leq k$ **do**
- 6: $\mathbf{y}_i :=$ Trajectory point end-effector position
- 7: **if** $i = 0$ **then**
- 8: $d_T = d_T + \|\mathbf{y}_i - \mathbf{x}\|$
- 9: **else**
- 10: $d_T = d_T + \|\mathbf{y}_i - \mathbf{y}_{i-1}\|$
- 11: **end if**
- 12: $i = i + 1$
- 13: **end while**
- 14: **if** $d_T \leq (1 + m_d) * d_0$ **then**
- 15: $s = \text{true}$
- 16: **else**
- 17: $s = \text{false}$
- 18: **end if**
- 19: **Return** s

We have implemented a planned motion checker to avoid this kind of undesirable behavior (Algorithm 3). When the motion planner calculates a trajectory for the robot, the role of the planned motion checker is to validate it. The distance from the starting and target position of the end-effector is compared to the total distance travelled along the

points that the planner has calculated for the trajectory. If the deviation is greater than a specified value, the planner is commanded to calculate a new trajectory; otherwise, the trajectory is taken as valid. Recalculation of the trajectory is performed a given number of times, if all of them result in defective trajectories the execution is interrupted awaiting user confirmation. This is a fail-safe mechanism that allows our cell to be suitable for application in an industrial environment. We will set a maximum admissible deviation parameter m_d that is dependent on the motion to be executed; in most cases, setting a maximum deviation 20% is enough to obtain safe trajectories, but there are some cases (i.e., when the end-effector has to perform a considerable orientation change) when this value has to be manually set to a much higher value.

Figure 10 shows a comparison of two trajectories generated by the *MoveIt! Motion Planner* for the sole pickup task at the conveyor station. Both trajectories were generated using the same starting and target points, and both of them are seemingly valid. However, one of them follows almost a straight line, while the other performs a high deviation trajectory that makes the robot go outside the reasonable working space. Figure 10b shows the end-effector position for both of the calculated trajectories. The number of points in the planned trajectory, k , can vary greatly from one planned motion to another. At first, the number of trajectory points could seem like a way to distinguish between suitable and unsuitable trajectories, but checking the travelled distance has been proven to be much more robust and reliable. The distance between the starting point (red) and the target point (green) is 0.518 m. A suitable trajectory will try to follow the shortest path possible; in this case, it results in a path with a covered distance of 0.534 m, resulting in a 3% distance deviation. A high deviation trajectory finds another possible trajectory where the end-effector pose achieved is the same, but a much longer distance is travelled, 2.63 m corresponding to a 407% deviation. When a high deviation trajectory that exceeds the maximum permitted value is found, as in this second case, the checker flags it as unsuitable and planner is commanded to recalculate the trajectory.

5 Experiments

5.1 Grasping points calculation from the conveyor belt

The input to our algorithm (Algorithm 1) is a dense point cloud obtained from a digitizer. *GeoGrasp* can successfully calculate grasping points on the concave hull of an outsole in tens of milliseconds, while it could not calculate them in the original cloud due to the great amount of points it contained.

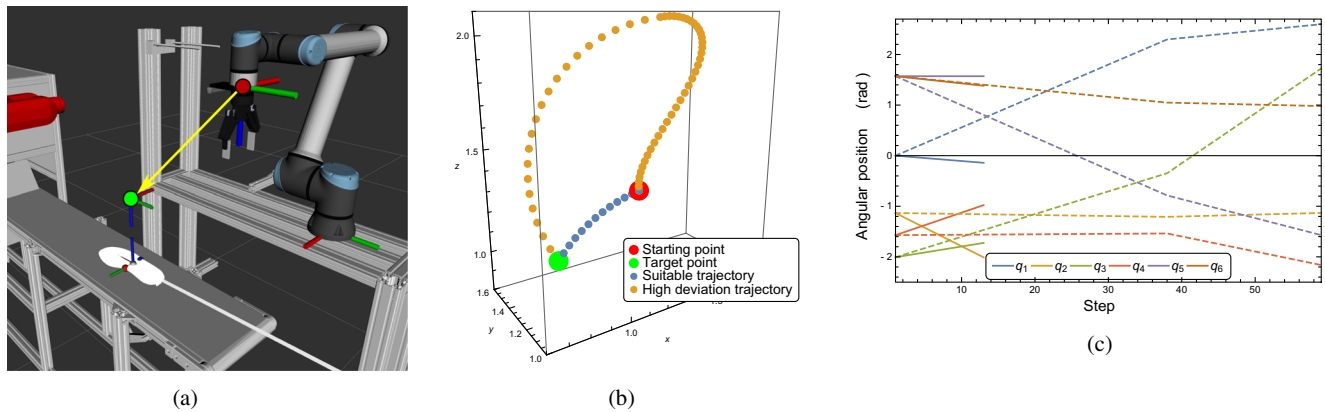


Fig. 10 Planned motion checking using Algorithm 3. **a** Motion to be planned. **b** 3D representation of trajectories. **c** Trajectory joint values. Solid lines: suit-able trajectory, dashed lines: high deviation trajectory

To check how good a representation of the real outsole this concave hull is, we perform a comparison between grasping points generated in three scenarios. In scenario 1, we calculate the grasping points for the concave hull of a outsole with $\alpha = 0.005$. For scenario 2, this concave hull is extended with its k-Nearest Neighbors ($k = 10$) using the PCL implementation based on FLANN [29]. The third scenario calculates grasping points on the concave hull correcting its center of mass and principal component to those of the original cloud computed using PCA. To do this, we have created a dataset with 20 sole models from commercial footwear. These soles are of various sizes and have different characteristics. We have chosen all of them from the right foot to be able to make a fairer comparison between them, but the method would work the same way if they were left foot soles.

First, we perform PCA on complete clouds, obtaining the centroid and first eigenvector of them. We can compare these values to those obtained from PCA on the concave hull and its extension containing the hull's 10-NNs. Figure 11 shows the mean and standard deviation of the centroid and first eigenvector with respect to the PCA of the original cloud, along with individual values for each of the outsoles, normalized to expected value (Eq. 3).

$$\left\| \frac{PCA(\mathbb{C}_s^{n \times 3}) - PCA(\mathbb{P}_s(\mathbb{C}_s^{n \times 3}))}{PCA(\mathbb{C}_s^{n \times 3})} \right\| \quad (3)$$

We can see a decrease in deviation of the hull cloud centroid with respect to the original one if we enhance the cloud with its nearest neighbors (NN), while the first eigenvector deviation remains practically the same. The improvement is not significant enough to compensate for the processing time required to look for 10-NNs of all of the points of the hull (Fig. 12), but we can try a different approach. Because we want to perform grasps along the contour of the sole, we can try enhancing grasping point calculation by correcting PCA values of the hull with those

of the original cloud when calculating the grasping points. This is the motivation of scenario three.

We can proceed to calculate the grasping points on soles for these three scenarios using our proposal of new *GeoGrasp* described in the previous section, and obtain the total elapsed time to obtain a result for the three scenarios: concave hull, enhanced concave hull with 10-NNs, and concave hull with PCA correction (i.e., using centroid and first eigenvector of original cloud when calculating grasping points on the hull). Total elapsed time includes all of the computations performed on the original cloud applicable for each scenario: hull generation, PCA, NNs, grasping point calculation, etc. Figure 12 shows mean values and standard deviations for the complete process of grasping point generation for all outsole samples, elapsed time is measured on a laptop with a 2014 Intel®Core™i7-4600U Processor.

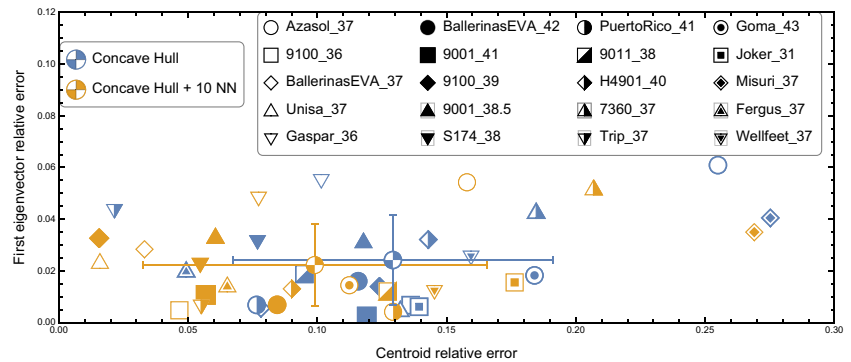
Overall, the most time-consuming process is nearest neighbor calculation on very dense point clouds, while concave hull generation is performed much faster. Adding PCA correction increases generation time, but includes characteristics from the original cloud to perform better grasps on the hull.

5.2 Grasping from rotation station

Grasping from the rotation station has a series of peculiarities that makes it a challenging task. Unlike the grasping station where we grasped the sole from the flat surface of the conveyor belt, the rotation station requires the gripper to perform an aerial grasp. The sole is lying on two metal rods that allow an inferior approach, which means that there is not a solid surface supporting the sole during the grasp.

The sole arrives to the rotation station directly from the grasping station and it is left lying on the rods. The positioning on the rods is precisely height controlled using the kinematics of the gripper and the height of the grasp obtained from the scanned point cloud.

Fig. 11 Relative differences of hulls with respect to PCA of original cloud computed from 20 sole models



The robotic arm performs a 180° rotation to reach the rotation station from below and takes a picture using the RGBD camera on its end-effector. Due to space limitations in the physical implementation, unlike the scanner point cloud (Fig. 13a), the point cloud obtained from the camera is not a complete view of the sole and lacks the shoe tip (Fig. 13b). This cloud is then segmented to keep only the section of the shoe between the metal rods, which is the place where the grasp can be performed. The grasping points are calculated using only this section of the sole (Algorithm 2).

If we consider the grasping points computed from a complete point cloud, G , as ground-truth (Algorithm 1), we can perform a comparison between the grasping points obtained from the complete view of the sole and those calculated with the partial view as RGBD, G' (Algorithm 2). We must perform some kind of matching between both point clouds. This is not a straightforward task for two main reasons. First, they have been obtained from different sensors with different resolutions; in the grasping station a laser scanner is used, while in the rotation station we use a RGBD camera. Second, the point clouds are of two different views of the sole; the laser scanner obtains a top view of the sole, while the RGBD camera records a bottom

view. The upper and bottom sides soles have very different characteristics making feature extraction unsuitable for matching.

We opted for an automated matching procedure based on two affine transformations. First, we perform a translation to bring together the end points of the sole's heel and a rotation to align the position of the sole (Fig. 14a). This initial matching is then adjusted obtaining the optimal affine transformation between five corresponding points of the sole present in both point clouds (Fig. 14b); this method uses singular value decomposition (SVD) to minimize the least-square error between the point sets [30].

Once the matching is done, we can compare the grasping points obtained at the conveyor grasping station $G = \{g_1, g_2\}$ with those of the rotation station $G' = \{g'_1, g'_2\}$. This can give us an idea of how close they are to each other, and see the difference it makes to use the whole sole or only the central part of it to calculate the grasping points. This will be more noticeable when the sole has a non-standard shape or a relatively prominent heel. Although our dataset is composed of 20 soles, Fig. 15 only shows the error obtained in 13 of the 20 soles shown in Fig. 11, made with material of low and medium flexibility. This error is calculated as the root mean square error (RMSE) between the grasping points in both scenarios (Eq. 4).

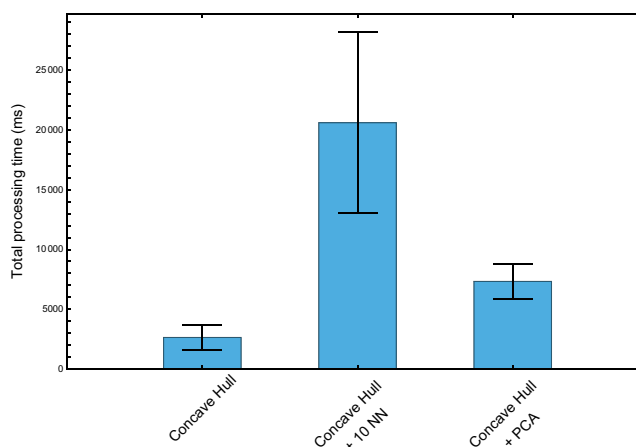
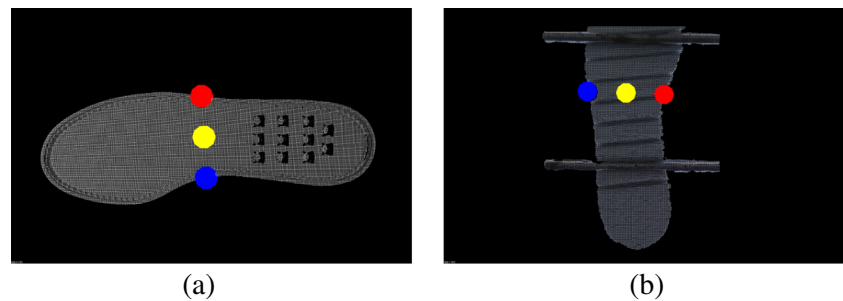


Fig. 12 Total elapsed time for grasp generation

$$\sqrt{\frac{1}{2} \sum_{i=1}^2 \|g_i - g'_i\|^2} \quad (4)$$

The error analysis shows that the matching error is greater in soles with heel and there is no significant difference between low and medium flexibility. In this experiment, we have preferred to show only low and medium flexibility errors because high-flexibility soles sometimes cause dynamic problems in the grasping task from the rotation station, which will be discussed in the next section. Nevertheless, it is possible to note that our algorithm to compute grasping point on soles works well with any type of sole (Fig. 16), including not only low and medium flexibility but also full flexibility.

Fig. 13 Different point grasping computed from upper view from scanner and bottom view from RGBD camera



5.3 From model to production

Here, we tested our setup for the entire dataset of soles with varying sizes, colors, and shapes, repeating the process 10 times for each one. Therefore, 200 tests were carried out. All shoe soles are made of materials with some level of flexibility, to allow the natural actions of movement of the human foot. In our experimentation, we have classified our sole samples into three categories: low, medium, and full flexibility. This measure of flexibility accounts for the degree of deformation suffered by the object when grasped under the action of gravity. Low flexibility refers to minimum deformation, almost unnoticeable, while medium flexibility accounts for a slight bend of the sole. In the full flexibility group, all the soles experience a considerable deformation that changes its original shape.

From the results shown in Table 3, we can see that low and medium flexibility soles work well with our cell. Soles that are very flexible and change their shape when grasped are highly likely to slip when grasped in the rotation station. The rotation station requires an aerial grasp, which is very difficult to perform if the sole changes its shape and there is no deformation control in the system.

We can go further in our analysis and see how the presence of a heel affects the process in case of low and medium flexibility soles, classifying them further into soles with

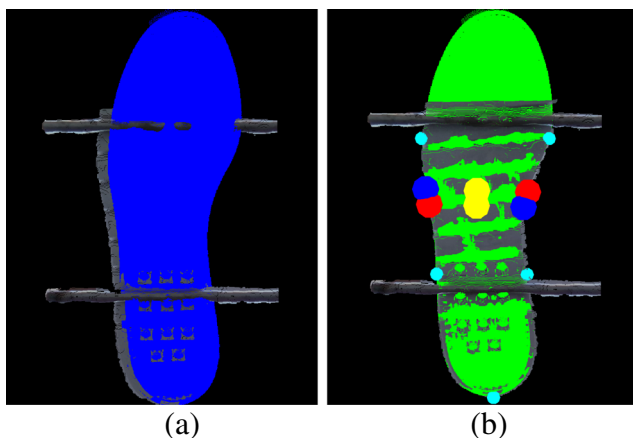


Fig. 14 Matching of scanner top view and camera bottom view. **a** First step. **b** Second step

and without heel. The results in Table 2 show that the absence of heel produces a very high success rate, while heels have a great impact on the grasping performed in the second station. Heels change the geometric characteristics of the sole and are difficult to represent by the point cloud when only an upper or lower view of the sole is present. Point clouds obtained by laser sensors and RGBD cameras record only points on the surface; they are not able to give a representation of the volume of the object without further post-processing. From the data in Fig. 15, we can confirm that heel presence has a considerable impact on the location of the centroid of the object.

We have obtained the measurement accuracy for grasping points as the positioning error in the conveyor and rotation stations. We have calculated this value for all of the trials performed using outsoles without heels (10 soles with 10 trials per sole), representing it as the RMSE between the position of grasping points computed on the point cloud and those points in which the end-effector grasps the objects.

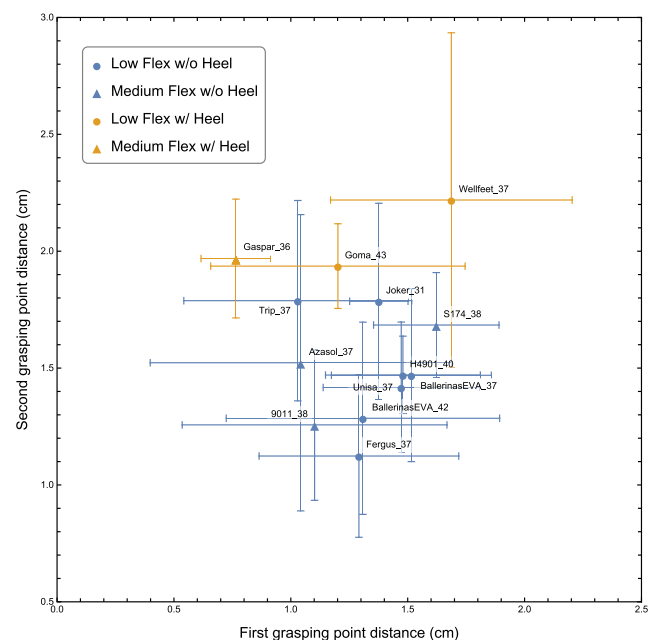


Fig. 15 Distance of grasping points from conveyor to rotation station

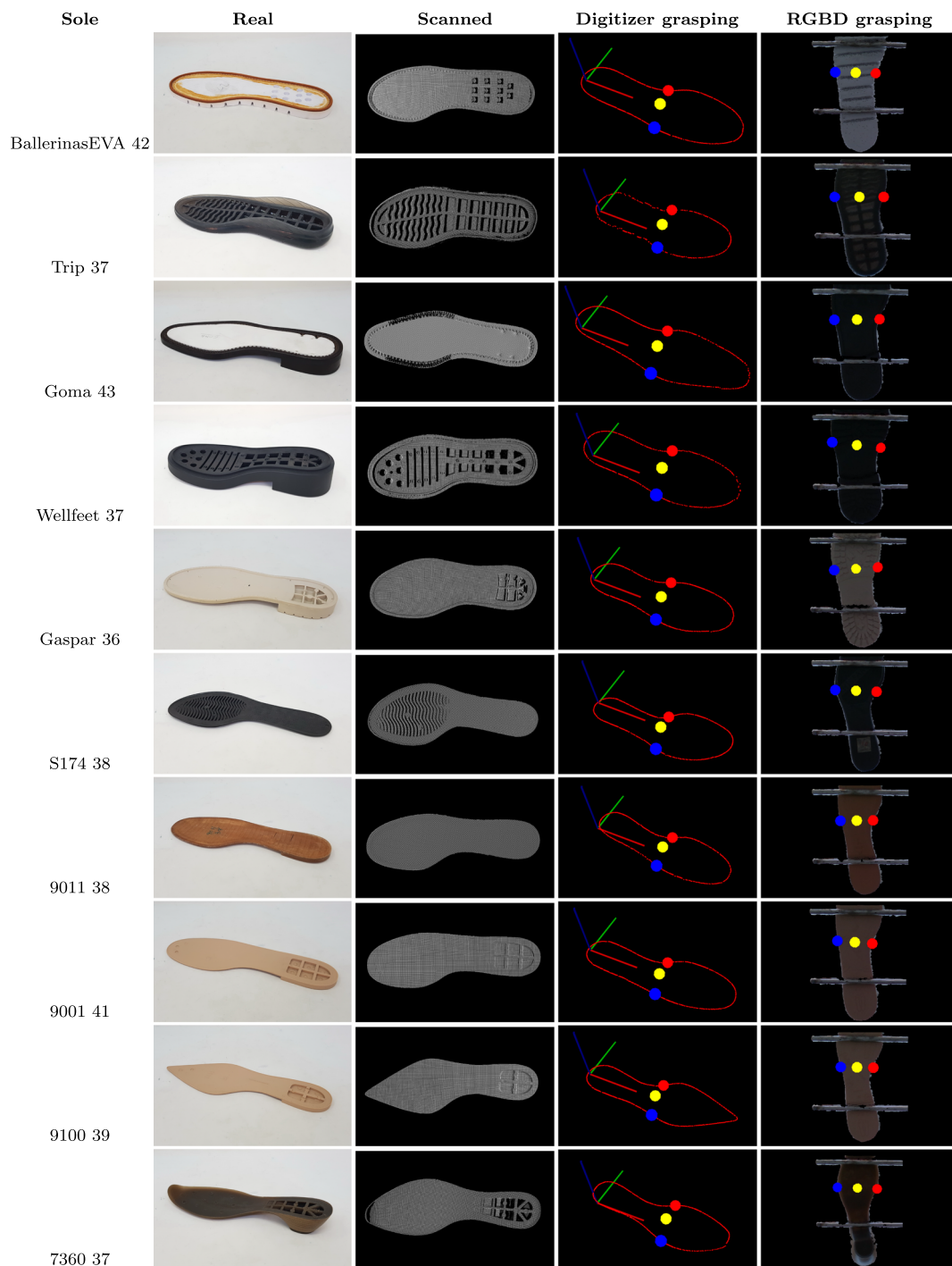


Fig. 16 Several examples of our shoe sole dataset

Table 2 Grasping success ratio for soles according to flexibility level

	Low	Medium	Full
No. of soles	9	4	7
Conveyor	98.9%	100%	75.7%
Rotation	84.3%	85%	34%

Table 3 Grasping success ratio for low and medium flexibility soles classified according to the presence of heel

	No heel	Heel
No. of soles	10	3
Conveyor	100%	96.7%
Rotation	95%	48.3%

The value for the conveyor station is 1.5 mm, while the value for the rotation station is 3.9 mm. As shown in Table 3, these values obtain 100% and 95% success rates, showing that sole typology (flexibility and heel presence) is a bigger determinant in grasping success than positioning error. We can conclude that our method and the designed cell prototype can be recommended to be used with low and medium flexibility soles without heels. Future work will include end-effector force control to improve grasping success for the different sole typologies.

Once the whole process for one sole was tested correctly, we implemented queues to make the process more efficient. Several soles can be present on the conveyor and their scanned models are stored in a queue. These soles are processed as they arrive to the pickup zone. While the robot arm is performing the tasks for rotation and assembly station, more soles are being processed at the scanner until one reaches the pickup zone. When the robot arm has finished the process, it returns to the pickup zone to grasp the next sole and the process starts again. The usage of queues endows for an asynchronous processing of soles and reduces idle times.

Figure 17 shows some frames of the grasping tasks in our real and virtual sole assembling robotic workcells for footwear manufacturing: the first frame illustrates sole grasping from conveyor pickup zone, the second frame presents sole drop on the rotation station, and the third frame shows sole grasping from below the rotation station.

We are using a Local Area Network (LAN) with four different physical nodes: an Ubuntu computer with the main process, a Windows box with the laser scanner digitizer software, the UR10 robot, and a Raspberry Pi with the RGBD camera. The main node is an Ubuntu 16.04 box with a 2019 Intel®Core™i7-9700F and 16 GB of RAM. At idle state, the process requires 1040 MB of RAM. The

network usage is 60 KB/s because the robot position is being broadcast continuously to the main node. Each digitized cloud from the conveyor takes up 6 MB in the queue. Considering the distance between the beam of light of the digitizer and the pickup zone, there could be a maximum of 4 soles simultaneously in the queue before the first one is processed. In the rotation station, the complete scene obtained from the RGBD camera is 28 MB, but only one scene is processed at a time. When the process is running, network usage can increase to several MB/s considering point cloud transmission between nodes.

Processing time of the complete process is not very dependent on the type of sole to be processed. Larger soles result in bigger point clouds that need more time to be processed in the conveyor station (i.e., concave hull generation and PCA feature extraction). As shown in Fig. 12, processing time for grasping point calculation is in the order of seconds for the conveyor station, but because there are intermediate processes (glue application and conveyor motion) between the scanning and the picking up, it does not really influence the overall processing time (Fig. 4). For the rotation station, no hull must be generated making the process much faster, the time needed for *GeoGrasp* to process the sole and obtain grasping points is in the order of milliseconds. Complete sole processing by the robot takes around 50 s: this includes from sole pickup in conveyor station to assembly station dropping and returning to starting position. With the introduction of queues, while the robot is performing the required motion with the sole, the conveyor is turned on and the laser scanner continues processing soles until one reaches the pickup position.

A flowchart of the final prototype implemented is shown in Fig. 18, where the three independent tasks running in parallel are shown. The first task processes soles on the conveyor as they arrive to the scanner, adding their digitized

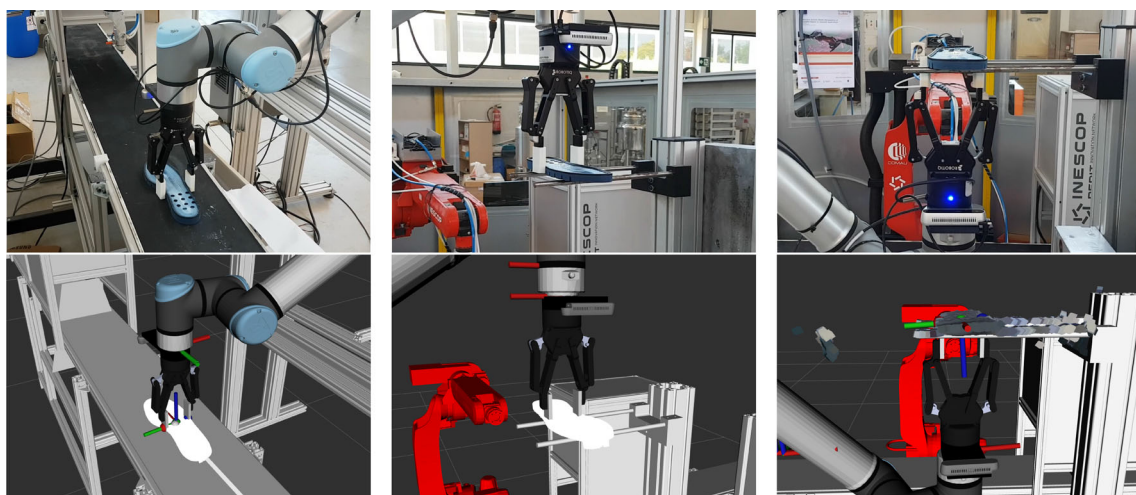
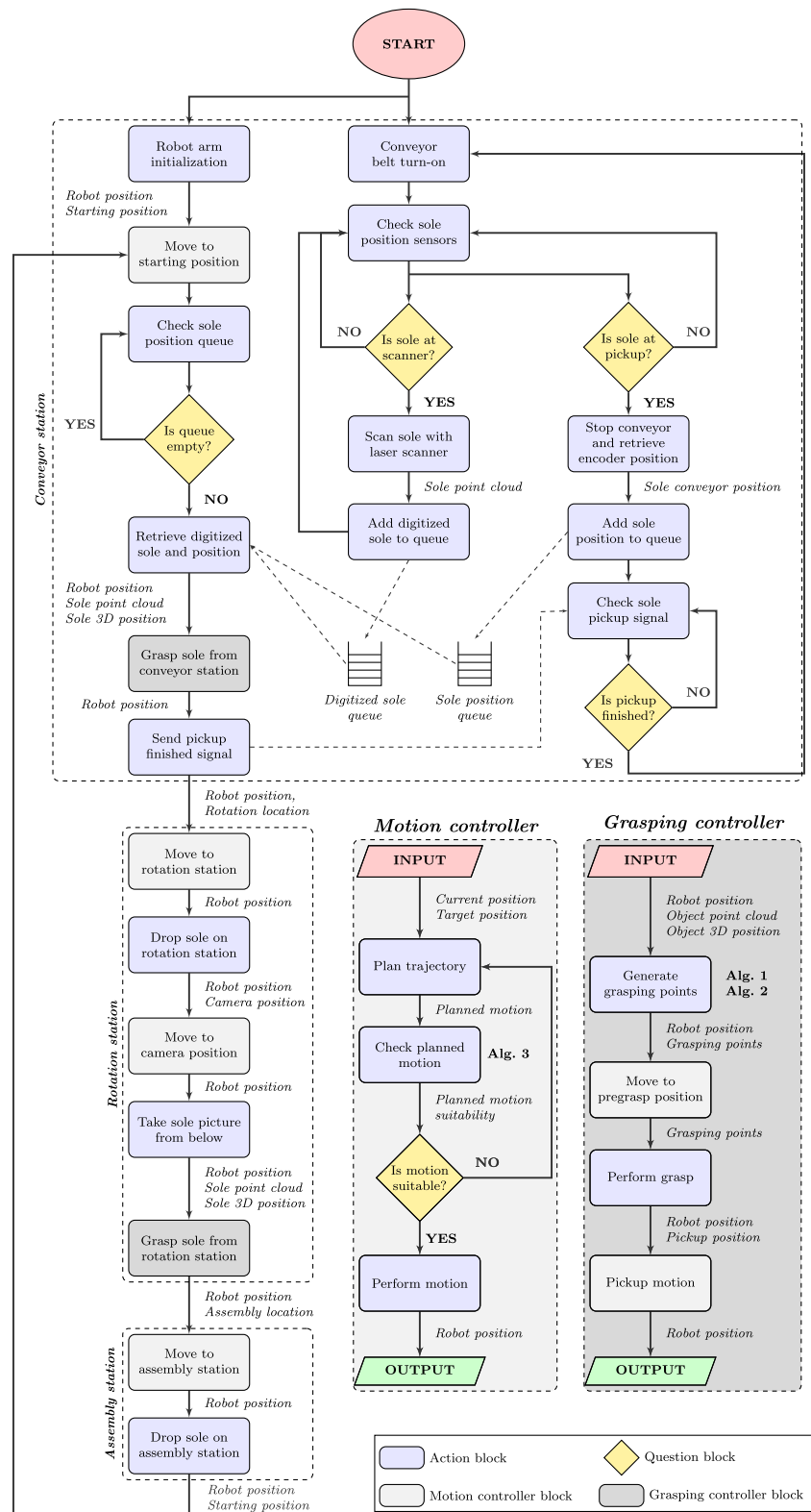


Fig. 17 Grasping tasks in our proposal of conveyor and rotation station: real workcell versus virtual scenario

Fig. 18 Flowchart of the overall operation of the implemented proposal. Three independent tasks run at the same time: sole scanning, sole pickup positioning, and robotic arm controlling. Algorithms 1 and 2 are used in grasping controller blocks for grasping points calculation and Algorithm 3 is used in the motion controller to check planned motions



versions into a queue. The second task is waiting for a sole to reach the pickup zone on the conveyor and once this happens it stops the conveyor motion, populating the position queue with the sole current location. The first and

second tasks are set on hold when the conveyor stops, waiting for the sole at the pickup zone to be picked up by the robotic arm. The third task controls robot arm operation and it is continuously processing soles that arrive to the pickup

zone. Once the sole is picked up, the conveyor is set on motion and tasks 1 and 2 resume. The robot arm task will process the sole through all the stations and return to the starting position to pick up the next sole that should be ready at the pickup zone.

6 Conclusions

In this work, we have proposed important improvements in *GeoGrasp*, introducing an algorithm based on concave hull generation. In this way, we can quickly obtain the outsole's contour and compute grasps overcoming the limitations of *GeoGrasp* working with dense clouds. We have tested three scenarios of grasping points generation on concave hulls: without modifications, enhancing the hull with its 10-NNs, and using PCA correction. We can improve grasping points computation performing a PCA on the original point cloud and correct the hull's characteristics, i.e., centroid and first eigenvector, with those of the original cloud. Concave hull enhancing with PCA correction provides information about the typology of the outsole, providing data about its density and interior features that has been lost when generating the hull; therefore, it can be used to obtain grasping points that are better aligned with the original point cloud characteristics.

In the real-world application, we have studied two grasping procedures with their own peculiarities. First, once the sole is scanned, we pick it up from the conveyor belt. In this case, a concave hull is obtained to reduce the grasping point generation time, correcting it with the PCA values of the original cloud. This sole is taken to a second station where an aerial grasp from below must be performed. We use an RGBD camera on the end-effector to obtain a partial point cloud of the sole and calculate grasping points. To produce safe robot motion, we have implemented a planned motion checker to restrict the deviation in the trajectory of the end-effector when calculating point-to-point motion.

There are some limitations to this method. One of the main disadvantages of 2D hull generation is information loss about the interior characteristics of the object. This is not an issue if the sole has a uniform thickness all around, but it can be an issue if the sole has different heights (e.g., a high heel) or non-symmetric internal features. This inconvenience has been partially mitigated using PCA correction. The sole's characteristics have a significant impact on the performance of the method. Picking up soles from the conveyor has a very high success ratio overall, while soles with a very high flexibility have a low success when performing the aerial grasp necessary for the rotation station. The presence of heel also affects the performance of the aerial grasp, due to the fact that a partial lower view is used to calculate the grasping points in the rotation station.

Heel presence modifies the location of the centroid of the sole and it is impossible to notice without further post-processing in the cloud obtained from the RGBD camera.

In conclusion, the current implementation of our method and the designed cell prototype can be recommended to be used with low and medium flexibility soles without heels. We plan to mount force sensors on the gripper to perform force control on outsole grasping and expect to improve grasp success on all sole typologies. Future research lines include tackling outsole manipulation, using a dual-robot setup to grasp the sole from both ends in the rotation station and actively control its deformation while pressing it together with the finished upper.

Supplementary information The online version contains supplementary material available at (<https://doi.org/10.1007/s00170-021-06697-0>).

Acknowledgements Thanks to INESCOP Footwear Technological Institute (<http://www.inescop.es/>), Polígono Industrial Campo Alto, Elda, Alicante (Spain). INESCOP provided soles and know-how about shoe making.

Author contribution P. Gil and F. Torres conceptualized the proposal. J.F. Gómez was responsible for describing traditional issues of footwear manufacturing and designing the fabrication cell. Then, G. Oliver implemented the virtual fabrication cell in ROS and, more later, all the authors implemented the physical fabrication cell. The robotic grasping algorithm was designed and implemented by G. Oliver and P. Gil. The experimental methodologies, formal analysis, and results validation were carried out by P. Gil, F. Torres, and G. Oliver. Finally, G. Oliver and P. Gil were responsible for writing the draft and F. Torres and F.J. Gómez supervised the work. All the authors reviewed and edited the final document.

Funding Research work was completely funded by the European Commission and FEDER through the COMMANDIA project (SOE2/P1/F0638), supported by Interreg-V Sudoe. Part of the facilities used were provided by the Footwear Technological Institute (INESCOP).

Availability of data and materials The authors confirm that the data supporting the findings of this study are available within the article and/or its [Supplementary Materials](#). The raw data that support the findings of this study are available from the corresponding author, P. Gil, upon a reasonable request.

Code availability Custom code developed is considered industrial intellectual property and is not to be released to the public domain. Individual software applications can be requested to the corresponding author, P. Gil, upon a reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest nor competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate

if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Luximon A, Luximon Y (2009) Shoe-last design innovation for better shoe fitting. *Comput Ind* 60(8):621–628. <https://doi.org/10.1016/j.compind.2009.05.015>
- Jimeno-Morenilla A, García-Rodríguez J, Orts S, Davia-Aracil M (2016) GNG Based foot reconstruction for custom footwear manufacturing. *Comput Ind* 75:116–126. <https://doi.org/10.1016/j.compind.2015.06.002>
- Crabtree P, Dhokia VG, Newman ST, Ansell MP (2009) Manufacturing methodology for personalised symptom-specific sports insoles. *Robot Comput Integr Manuf* 25(6):972–979. <https://doi.org/10.1016/j.rcim.2009.04.016>
- Yang Y-I, Yang C-K, Chu C-H (2014) A virtual try-on system in augmented reality using RGB-d cameras for footwear personalization. *J Manuf Syst* 33(4):690–698. <https://doi.org/10.1016/j.jmsy.2014.05.006>
- Cicconi P, Russo AC, Germani M, Prist M, Pallotta E, Monteriù A (2017) Cyber-physical system integration for industry 4.0: Modelling and simulation of an induction heating process for aluminium-steel molds in footwear soles manufacturing. In: *Proc. of IEEE 3rd Int.forum on research and technologies for society and industry (RTSI)*, Modena, pp 1–6. <https://doi.org/10.1109/RTSI.2017.8065972>
- Maurtua I, Ibarburen A, Tellaeche A (2012) Robotic solutions for footwear industry. In: *In Proc. of IEEE 17th int. conf. on emerging technologies and factory automation (ETFA)*, Krakow, pp 1–4. <https://doi.org/10.1109/ETFA.2012.6489780>
- Lai M-Y, Wang L-L (2008) Automatic shoe-pattern boundary extraction by image-processing techniques. *Robot Comput Integr Manuf* 24(2):217–227. <https://doi.org/10.1016/j.rcim.2006.10.005>
- Caruso L, Russo R, Savino S (2017) Microsoft Kinect V2 vision system in a manufacturing application. *Robot Comput Integr Manuf* 18:174–181. <https://doi.org/10.1016/j.rcim.2017.04.001>
- Hu Z, Marshall C, Bicker R, Taylor P (2007) Automatic surface roughing with 3D machine vision and cooperative robot control. *Robot Auton Syst* 55(7):552–560. <https://doi.org/10.1016/j.robot.2007.01.005>
- Wu X, Li Z, Wen P (2017) An automatic shoe-groove feature extraction method based on robot and structural laser scanning. *Int J Adv Robot Syst* 14(1):1–14. <https://doi.org/10.1177/1729881416678135>
- Kim J.-Y. (2004) CAD-Based automated robot programming in adhesive spray systems for shoe outsoles and uppers. *J Robot Syst* 21(11):625–634. <https://doi.org/10.1002/rob.20040>
- Pagano S, Russo R, Savino S (2020) A vision guided robotic system for flexible gluing process in the footwear industry. *Robotics and Computer-Integrated Manufacturing* 65. <https://doi.org/10.1016/j.rcim.2020.101965>
- Choi H, Hwang G, You S (2008) Development of a new buffing robot manipulator for shoes. *Robotica* 26(1):55–62. <https://doi.org/10.1017/S026357470700358X>
- Jatta F, Zanorri L, Fassi I, Negri S (2004) A roughing/cementing robotic cell for custom made shoe manufacture. *Int J Comput Integr Manuf* 17(7):645–652. <https://doi.org/10.1080/095112042000273212>
- Nemec B, Žlajpah L (2008) Robotic cell for custom finishing operations. *Int J Comput Integr Manuf* 21(1):33–42. <https://doi.org/10.1080/09511920600667341>
- Gracia L, Perez-Vidal C, Mronga D, Paco J-M, Azorin J-M, Gea J (2017) Robotic manipulation for the shoe-packaging process. *Int J Adv Manuf Technol* 92:1053–1067. <https://doi.org/10.1007/s00170-017-0212-6>
- Tsai Y-T, Lee C-H, Liu T-Y, Chang T-J, Wang C-S, Pawar SJ, Huang P-H, Huang J-H (2020) Utilization of reinforcement learning algorithm for the accurate alignment of a robotic arm in a complete soft fabric shoe tongues automation process. *J Manuf Syst* 56:501–513. <https://doi.org/10.1016/j.jmsy.2020.07.001>
- Kim M, Kim J, Shin D, Jin M (2018) "Robot-based shoe manufacturing system. In: *Proc. of 18th int. conf. on control, automation and systems (ICCAS)*, Daegu/wallyeong, pp 1491–1494
- Rocha LF, Veiga G, Ferreira M, Moreira AP, Santos V (2014) Increasing flexibility in footwear industrial cells. In: *Proc. of IEEE int. conf. on autonomous robot systems and competitions (ICARSC)*, Espinho, 291–296
- Belman-Lopez CE, Jimenez-Garcia JA, Hernandez-Gonzalez S (2020) Comprehensive analysis of design principles in the context of Industry 4.0. *Revista Iberoamericana de Automática e Informática Industrial* 17:432–447. <https://doi.org/10.4995/riai.2020.12579>
- Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler RC, Ng AY (2009) ROS: an open-source robot operating system. In: *Proc. of int. conf. on robotics and automation (ICRA)*, Kobe
- Coleman D, Şucan IA, Chitta S, Correll N (2014) Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study, vol 5. <https://doi.org/10.6092/JOSER.2014.05.01.p3>
- Denavit J, Hartenberg RS (1955) A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J Appl Mech* 23:215–221
- Zapata-Impata BS, Gil P, Pomares J, Torres F (2019) Fast geometry-based computation of grasping points on three-dimensional point clouds. *Int J Adv Robot Syst* 16(1):1–18. <https://doi.org/10.1177/1729881419831846>
- Rusu RB, Cousins S (2011) 3D is here: point cloud library (PCL). In: *2011 IEEE int. conf. on robotics and automation, shanghai*, pp 1–4. <https://doi.org/10.1109/ICRA.2011.5980567>
- Edelsbrunner H, Kirkpatrick DG, Seidel R (1983) On the shape of a set of points in the plane. *IEEE Trans Inf Theory* 29(4):551–559. <https://doi.org/10.1109/TIT.1983.1056714>
- Barber CB, Dobkin DP, Huhdanpaa H (1996) The Quickhull algorithm for convex hulls. *ACM Trans Math Softw* 22(4):469–483. <https://doi.org/10.1145/235815.235821>
- Oliver G, Gil P, Torres F (2020) Robotic workcell for sole grasping in footwear manufacturing. In: *2020 25th IEEE International conference on emerging technologies and factory automation (ETFA)*, Vienna, Austria 2020
- Muja M, Lowe DG (2009) Fast approximate nearest neighbors with automatic algorithm configuration. In: *Int. conf. on computer vision theory and applications (VISAPP)*, Lisboa, pp 331–340. 10.1.1.160.1721
- Arun KS, Huang TS, Blostein SD (1987) Least-Squares Fitting of two 3-D point sets. *IEEE Trans Pattern Anal Mach Intell* 9(5):698–700. <https://doi.org/10.1109/TPAMI.1987.4767965>

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