Document downloaded from:

http://hdl.handle.net/10251/151046

This paper must be cited as:

Gracia Calandin, LI.; Perez-Vidal, C.; Mronga, D.; Paco, JD.; Azorin, J.; Gea, JD. (2017). Robotic manipulation for the shoe-packaging process. The International Journal of Advanced Manufacturing Technology. 92(1-4):1053-1067. https://doi.org/10.1007/s00170-017-0212-6

The final publication is available at

https://doi.org/10.1007/s00170-017-0212-6

Copyright Springer-Verlag

Additional Information

Noname manuscript No.
(will be inserted by the editor)

Robotic manipulation for the shoe packaging process

Luis Gracia* · **Carlos Perez-Vidal** · **Dennis Mronga** · **Jose-Manuel de Paco** · **Jose-Maria Azorin** · **Jose de Gea**

Received: September 23, 2016 / Accepted: date

Abstract This paper presents the integration of a robotic system in a human-centered environment, as it can be found in the shoe manufacturing industry. Fashion footwear is nowadays mainly handcrafted due to the big amount of small production tasks. Therefore, the introduction of intelligent robotic systems in this industry may contribute to automate and improve the manual production steps, such us polishing, cleaning, packaging and visual inspection. Due to the high complexity of the manual tasks in shoe production, cooperative robotic systems (which can work in collaboration with humans) are required. Thus, the focus of the robot lays on grasping, collision detection and avoidance, as well as on considering the human intervention to supervise the work being performed. For this research, the robot has been equipped with a Kinect camera and a wrist force/torque sensor so that it is able to detect human interaction and the dynamic environment in order to modify the robot's behavior. To illustrate the applicability of the proposed approach, this work presents the experimental results obtained for two actual platforms, which are located at different research laboratories, that share similarities in their morphology, sensor equipment and actuation system.

Keywords Robotic manipulation · shoe industry · human-robot cooperation · dynamic trajectory planning

L. Gracia (*Corresponding author)
Instituto IDF, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

C. Perez-Vidal and J.M. Azorin
Miguel Hernandez University (UMH), Elche, Spain

D. Mronga and J. de Gea
German Research Center for Artificial Intelligence (DFKI), Bremen, German

J.M. de Paco
The European Organization for Nuclear Research, CERN, Geneve, Switzerland

1 Introduction

The use of robotic arms in industry, even at hyper-flexible productions lines, is being favored due to new devices, sensors and algorithms. However, in the footwear industry, robots are still restricted to very few tasks. In fact, not many examples of robotic applications can be found in shoe manufacturing industry [1] [2] [3] [4]. One of the reasons is the still not overcome difficulties in handling flexible materials. However, some projects have been developed to increase the productivity of specific steps in shoe manufacturing process *using robotics*. Next, these projects are reviewed.

The EuroShoe project develops an innovative robotic cell [5] that is able to perform finishing operations in shoe manufacturing, such as cleaning or polishing. It uses a robotic manipulator equipped with a force controlled head that works with a CAD-based software, which enables the robot to optimally adjust trajectories to the contours of the shoe. Furthermore, an innovative three-fingered gripper [6] has been developed within this project in order to adapt to the materials commonly used in shoe manufacturing.

The INTELISHOE project [7] had the general goal of reducing the time-to-market in SMEs of traditionally handcrafted goods such as footwear by taking into account all available technologies. A distributed design concept and prototypes have been implemented in this project.

The SSHOES project [8] addresses the implementation of new adaptive production processes for footwear and insoles, which also includes the development of robotic demonstrators and 3D design tools. Whereas, researchers from Minho University present in [9] a software application for optimizing shoe sole halogenation and lead roughing processes, two steps commonly involved in the footwear manufacturing.

The CEC-MADE-SHOE project [10] has recently developed advanced tools for the customization process (magic mirror). The FIT4U project [11] aims at responding to the growing demand for consumer oriented product customization by conceiving an Engineering Framework, meant as the set of tools and manufacturing technologies required to obtain a consumer centered product in sport footwear.

Within the European project RoboFoot [12], manipulation strategies for non-rigid objects, programming tools, and sensor-based control approaches have been developed in order to overcome the complexity of automating the shoe production processes. Regarding 3D modelling, the work described in [13] uses object-oriented CAD systems for designing heels and insoles.

Table 1 shows a summary of projects funded by the E.U., specifying starting and ending dates, project IDs, funding program, total cost of the project and number of participants.

Among the research projects discussed above only a few of them are using a robot system as main device. For those cases, *motion planning* has to be performed to find a path from the current robot configuration to the desired one, eventually taking into account constraints like joint limits or collision avoidance with the obstacles in the environment. Regarding motion planning

Table 1 Summary of projects funded by the E.U.

Project Name	Dates (Start-End)	ID and Program	Budget (EUR)	Num. of Partners
EuroShoe	2001/03/01 - 2004/06/30	G1RD-CT-2000-00343 FP5-GROWTH	17.384.308	33
INTELISHOE	2000/12/01 - 2002/05/31	IST-1999-20949 FP5-IST	1.036.080	15
SSHOES	2009/07/01 - 2012/06/30	229261 FP7-NMP	4.874.025	11
CEC-MADE-SHOE	2004/10/01 - 2008/09/30	507378 FP6-NMP	20.417.201	16
FIT4U	2009/07/01 - 2012/06/30	229336 FP7-NMP	5.709.500	13
ROBOFOOT	2010/09/01 - 2013/02/28	260159 FP7-ICT	3.685.073	10

in static or slowly changing environments, many solutions can be found, e.g., sampling based planners [14][15] or gradient optimization methods [16]. However, the application of robots in human-centered environments requires the robot system to react to dynamic changes, which means that the generated trajectories have to be adapted online in response to sensory feedback. Basically, there are two possibilities to achieve it, either by locally modifying the off-line generated trajectory using a reactive controller or by generating online a new complete solution. The advantage of the former is the computational speed, since replanning is avoided, and its main disadvantage is that the continuous modification of the path may lead the robot into a local minima. An example for the reactive path modification technique is the Elastic Strips Framework [17], which uses virtual force fields originated by the obstacles in the environment to incrementally modify a given trajectory in the task space. Similarly, the Reflexxes Motion Library [18] provides tools and methods for smoothly modifying a trajectory in the joint space and reacting on sensory events. Whereas, path replanning techniques try to update online a solution for a given path planning problem. An example of this approach is the Anytime Path Planning framework [19], which constantly monitors the available sensor input and updates the plan in case that a change is detected. Furthermore, the Lightning Planning Framework [20] is able to learn from previously generated plans and adapt them to the new situation.

Additional considerations must be taken into account in unstructured cooperative environments in order to avoid collision between the robot and the human operator. In this case, the robot has to react to unforeseen events, which must be detected by the available *sensors*. In addition to the vision-based detection, force (and/or torque) sensors are usually incorporated to provide an extra safety measure.

Force (and/or torque) sensors are usually either integrated in the end-effector of the robot or at joint level. The latter case provides more sophisticated collision detection and compliance control at each joint. Regarding compliant robot behavior, active and passive approaches can be distinguished.

Active compliance can be achieved for intrinsically rigid actuators using a proper control concept in addition to force/torque sensing. Whereas, passive compliance is achieved with intrinsically compliant actuators and thus does not require force/torque sensing. Two examples of torque controlled robots that utilize these two approaches are the DLR lightweight arm LWR [21], which uses a sophisticated active compliance control, and the Barrett Whole Arm Manipulator (WAM) [22], which uses backdrivable cable drives and does not require any compliance control. Thus, the latter is not affected by the delays of the control loop but suffers low repeatability since the position control is less accurate with compliant actuators.

These force and backdrivablility considerations are a required to complement the vision system which globally monitors the robot's workspace, detects the obstacles and guides the end-effector to a goal position using, for instance, visual servoing techniques. For instance, a visual servoing method is presented in [23] for a robot working cooperatively with human operators. The avoidance of obstacles during object manipulation has been treated in many works, for example using a camera and SURF features [24]. In particular, in this work a depth camera (Microsof Kinect camera [25]) is used to generate a point cloud of the robot's environment. This information is used online to generate a motion plan and to avoid obstacles while interacting with the working scene.

In summary, this work presents the effort of several research projects carried out by the authors to achieve a higher level of automation in shoe packaging tasks. For this purpose, different setups (robots, robotic hands, software libraries, etc.) have been implemented and tested to evaluate the capabilities of robotic manipulation in the shoe industry.

The structure of the paper is as follows. Next section introduces the industrial application to be solved using a robotic system, while Section 3 presents the architecture (conceptual, hardware and software) proposed in this work to cope with it. Next, Section 4 develops the robot control system to properly perform the industrial task at hand. Then, the proposed approach is applied in Section 5 to two actual robotic platforms to show its feasibility and effectiveness. Subsequently, Section 6 summarizes the key findings of this research and the remaining challenges. Finally, some concluding remarks are given.

2 Description of the Application

2.1 Shoe Manufacturing

Compared to other industries (e.g., the automobile industry), the shoe industry has two main features that make it especially hard to automate. Firstly, a relatively big amount of tiny pieces are managed, which are difficult to handle for a robot. Secondly, there is a large number of products variants (e.g., $200 \cdot 6 \cdot 2 = 1200$, see description below), which will probably not be produced again due to fashion changes, with very short production runs (e.g., eight, see

description below). Hence, robotic platforms would have to be continuously re-programmed.

For this reason, fashion footwear production is currently mainly handcrafted. Some manufacturing processes are assisted by specialized machinery (last manufacturing, cementing, cutting, roughing, direct injection, etc.) and there exist highly automated lines in mass production of technical shoes (e.g. safety footwear). But most production is still handmade, being especially true in the case of high added value shoes production, where Europe and the USA maintain their leadership.

The main features of shoe manufacturing are:

- Large number of products *variants*:
  - ⋆ Models (typically around 200 per manufacturer): every year a minimum of two different collections (summer and winter) of shoes are manufactured and presented to customers, and will probably not be produced again due to fashion changes.
  - ⋆ Sizes (typically at least six) and sides (two): It is necessary to adapt each model to several sizes and two sides (left and right).
- Short production *runs*: eight pairs of shoes is the average order size in small and medium-sized enterprises.
- Complex *manufacturing* process. For each model, it is necessary to manufacture the last, to cut the leather, to fabric the parts and to produce the list of components: sole, heel, sock, strap, inner parts, etc.
- Complex *assembly* process. The assembly process is very laborious (around 80 different operations) and especially complex in fitting operations due to the non-uniformity and different elasticity of leather as well as the non-solid nature of the components that difficult their manipulation. Fig. 1 shows some of the painstaking and laborious tasks commonly performed in the shoe industry: leather cutting and splitting, sewing upper parts, leather pre-shaping, stitching the leather pieces to create the upper, nailing the insole on the last, mounting the upper over the last and outsole injection, the application of adhesive to join the outsole, etc.
- Manual *quality verification* (small spots, scratches, colour differences on the leather, correct alignment of pieces over the last, etc.)
- *Packaging*. This step is maybe the easiest and fastest one in the production line, but it is performed more than ten million times by a small-medium size shoe company.

Some of the activities performed to manufacture a shoe are extremely complex, and dexterous hands are required to get the desired quality level. But others could be automated and/or robotized. This is the case of upper roughing [26], final polishing [5], packaging and outsole mounting/gluing. This process of automation of the shoe industry could increase the competitiveness in this growing market, avoiding the migration of companies from Europe and USA to China, India or Vietnam, or even allowing them to come back again.

Fig. 2 graphically represents the sub-tasks involved on the packaging process. This production phase is one of the operations with higher workforce

Fig. 1 Steps performed to produce one shoe. Courtesy of SimplicityWorks Europe.

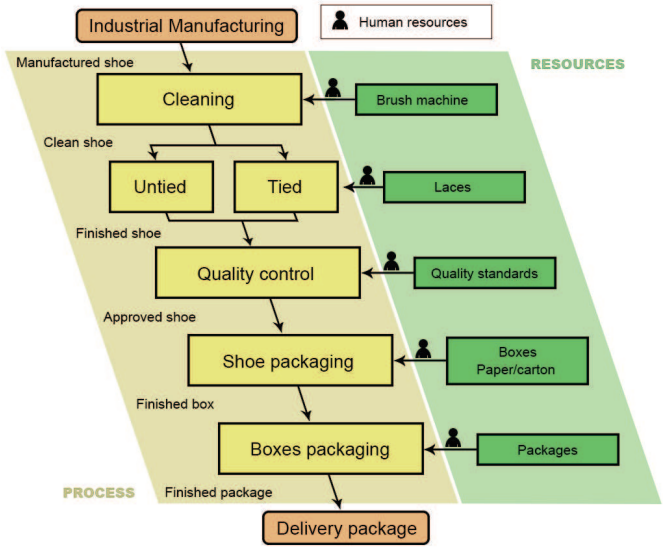

Fig. 2 Description of the packaging process

impact and its main sub-tasks are cleaning, quality control and putting the shoe inside the shoe box. For the shoe packaging sub-task, workers verify that both pairs match, they write the pair number on the box, the pair of shoes is put into the box (sometimes having to introduce some piece of paper and/or a plastic bag to separate and protect them) and the box is finally closed with the lid. This sub-task takes from 20 to 25 seconds to a human worker. Regarding

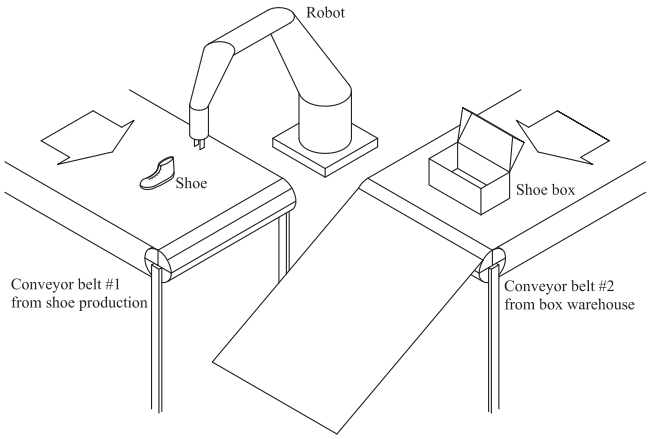

Fig. 3 Overview of the robot shoe packaging

the other sub-tasks, cleaning could be performed by a robot [27] meanwhile quality control is traditionally made by a human operator.

2.2 Robot System for Shoe Packaging

In this work, the shoe packaging process is addressed due to its special features. Automated shoe packaging has to deal with complex activities like robotic grasping, trajectory planning (including collision avoidance) and force control. Therefore, this work carries out a viability analysis of robot shoe packaging checking that the most critical tasks can be successfully tackled. Fig. 3 shows an overview of the activity to be developed, where two conveyor belts carry the shoe box and the pair of shoes to the robot workspace. When the pair of shoes is detected using computer vision, the robot takes them one by one placing them into the shoe box. Then, the robot closes the box and it is send to the next step.

In the shoe industry, the grasping position and orientation is essential to perform a stable grasping due to the rigid configuration of the outsole and deformability of the upper. Extreme soft parts of the shoe have to be avoided and, hence, grasp planning and tests have to be performed. Once the object is grabbed, the robot arm has to transport it from the conveyor belt to the shoe box, but some obstacles can be in the robot path. Therefore, obstacle avoidance is required in the control system to guarantee no collision. A trajectory planning strategy has to be designed to achieve this goal integrating force control as well. Force control considerations are introduced in this work to allow the interaction of a human worker inside the robot workspace for supervision and quality control tasks. The human worker could go inside the robot workspace or could modify the robot movement by handling the final effector.

3 Architecture

3.1 Conceptual architecture

Although every robot system has a specific architecture, most of them have been developed basically from two paradigms: reactive [28] and deliberative [29] (or hierarchical) paradigm. A combination of both approaches, known as hybrid [30] paradigm, is nowadays the most accepted one. A description of these three paradigms is given below.

A *paradigm* of a robot system is a mental model of how the robot operates. It can be described by the relationship between the three elementary functions: Sense, Plan and Act [30]; and establishes how sensory data is processed through the system to make decisions.

The *reactive* paradigm is inspired by animals' basic behaviors [28] and uses a Sense-Act organization: the robot has multiple instances of Sense-Act couplings, namely behaviors, which take the sensing data and independently compute the best action to carry out. Thus, the robot performs a combination of behaviors. Note that, there is no high-level intelligence for this case.

The *deliberative* paradigm uses a Sense-Plan-Act organization [29]: at each step the robot senses the environment to construct a world model, then plans (deliberates) the best action to carry out and, finally, it is executed.

The *hybrid* paradigm uses a Sense|Plan-Act organization [30]: firstly the robot plans how to best decompose the task into subtasks, then computes the suitable behaviors to accomplish each subtask and, finally, executes them as in the reactive paradigm. For this case, the sensing organization is a combination of reactive and deliberative cases: sensor data is used by each behavior that needs it, but it is also available to the planner to construct a world model.

This work proposes to use the hybrid architecture shown in Fig. 4 to solve the shoe packaging described in Section 2.2 with a robot system, where it can be seen the deliberative and the reactive parts of the architecture. Note that, three types of robot sensors are considered: camera, joint position sensors and force sensor.

In the deliberative part of the architecture, the data acquired by the camera is the input to the image processing algorithm (see Section 4.3) in order to detect the objects in the environment and to construct a world model. Subsequently, this information is used by the shoe grasping (see Sections 4.4 and 5.3) and the trajectory planning (see Sections 4.1 and 5.1) algorithms in order to generate a reference for the low-level robot control.

In the reactive part of the architecture, the data acquired by the force sensor is used to determine whether there is contact with the environment or not. In case that contact is detected (e.g., collision has occur between the robot and an object in the environment or the human operator is making some force in the robot end-effector) the robot task is immediately stopped and the robot control automatically switches to a force controlled mode, so that the operator is able to move the robot in the Cartesian space by pulling the end-effector. On the one hand, this reactive behavior prevents damage to the robot and

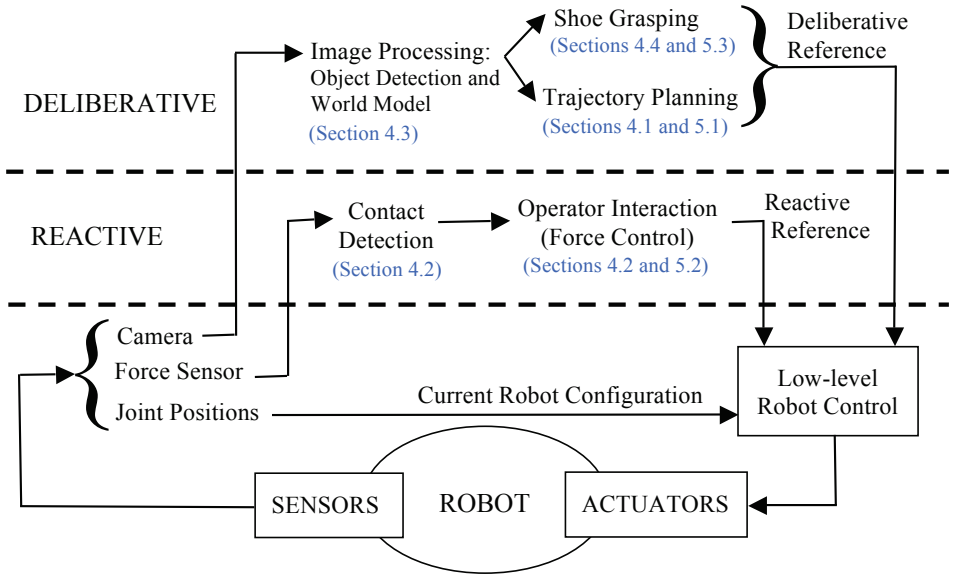

Fig. 4 Architecture used for this work

the objects in the environment and allows the human operator to relieve the system from a collision. On the other hand, this behavior allows the human operator to supervise the robot task. For instance, the operator can stop the robot motion while a shoe is being carried by the robot in order to visually inspect it and, if it is approved, the operator allows the robot to continue the task: introducing the shoe into the box, etc.

Finally, the low-level robot control computes the control signal for the actuators using the current robot configuration given by the joint position sensors and considering the deliberative or the reactive reference. In particular, if contact has been detected, the reference of the reactive part prevails over that of the deliberative part, as usual in hybrid robot architectures.

3.2 Hardware

This section aims at describing the hardware components used in the robotic platform, as well as giving a brief statement about its purpose for the overall system. Fig. 5 shows the main system components and the communication interfaces used between them.

Arm. The robot of both platforms used in Section 5 to test the proposed algorithms are based on Schunk modules. Both robotic arms are composed of seven joint modules of four different sizes (PRL120, PRL100, PRL080 and PRL060), with peak out torques ranging from 10 Nm to 372 Nm. Arms have a CAN bus line which links the modules with the control computer. In order

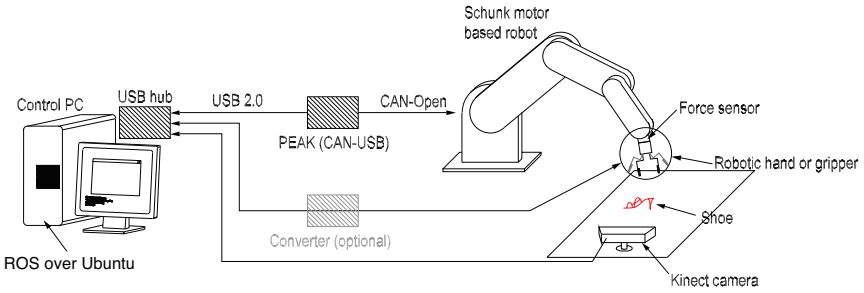

Fig. 5 Hardware architecture

to connect the arm (and hand) to the computer, a CAN-USB adapter is used. PCAN-USB (from Peak System manufacturer) enables simple connection and drivers are available for Linux Kernel 2.4 and higher.

Robot Hands. Three different grasping devices have been tested in this work for comparison and robustness purposes: the iCub hand from Italian Institute of Technology (IIT), the IH2 Azzurra Hand from Prensilia and a Schunk industrial gripper (servo electric 2-finger-parallel gripper type PG 70). It is interesting to remark that, initial tests for grasping algorithms were performed with industrial parallel grippers but, finally, the iCub (at DFKI Lab) and the IH2 Azzurra (at UMH Lab) hands were chosen to gain more dexterity.

3D Sensor. The Kinect camera is widely used in robotics due to its high performance and very low cost, e.g., about 150 USD. Furthermore, there are specific drivers like OpenNI and libraries like PCL (Point Cloud Library) prepared to work under ROS (Robot Operating System) [31] using 3D information extracted by a Kinect camera. The camera comprises a RGB camera, which stores three channel data in a 1280x960 resolution (color image capture), an infrared (IR) emitter and an IR depth sensor. The emitter emits infrared light beams and the depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This 3D sensor was selected, instead of a ToF camera, because the technology used by Kinect usually performs accurattely for indoor applications, only sharp angles or small structures could be difficult to characterize. Moreover, the minimum measuring distance for this 3D sensor is around one metre. Therefore, acquiring full frame depth measurements or considering an outdoor environment seem to be domain of ToF based depth measuring [32].

Control PC. ROS [31] allows distributed operation over multi-core, multiprocessor, GPUs, and clusters. The different tasks (vision, trajectory control and planning, etc.) could be executed over different computers in order to distribute the load of the system. In this case, only one computer has been

used with the following specifications: Intel Core i7 2600k, 4 cores, 3,40Ghz, 6Mb cache, Turbo Boost 2.0, RAM 8Gb, NVIDIA GeForce210 4Gb, HDD ST1000DL002-9TT153 ATA of 1Tb.

3.3 Software

The ROS platform [31] over Ubuntu has been chosen for this work. This platform was initially developed by Willow Garage Company in 2007 and has become a standard tool among robotics researchers and industry. It is similar in some aspects to other robot frameworks such as Player [33], YARP [34], OROCOS [35], CARMEN [36], ORCA [37], MOOS [38] and Microsoft Robotics Studio [39]. The ROS platform has been selected due to the availability of drivers for hardware devices and of libraries to help software developers to create robot applications using typical algorithms for trajectory planning, computer vision, robotic grasping, etc. Moreover, ROS provides the services of an operating system: hardware abstraction, low-level device control, etc. It is interesting to remark that ROS is the current trend in the most popular robots all over the world: the PR2 [40] of Willow Garage; the Care-O-bot [41] of Fraunhofer IPA; the partly AILA [42] at DFKI Robotic Innovation Center; among others.

In order to communicate with the CAN modules (arm Schunk modules and the robot hands) via the PCAN-USB interface, the ROS package *libpcan* [43], which wraps the PCAN drivers for Linux [44] and ROS [31], has been selected.

The ROS package *schunk_powercube_chain* [45] provides a configurable driver for a chain of Schunk powercube modules. A node [46] takes in joint velocities messages [47] being suscribed to the topic *command_vel (brics_actuator/ JointVelocities)* [48] and sends them to the hardware. The powercube chain can be initialized, stopped or recovered via ROS services [49]. Hardware configuration is done in the package *schunk_hardware_config* using a YAML file [50]. Some of the parameters configured in the YAML file are the name of the CAN module (e.g., PCAN), CAN device (e.g., /dev/pcan32), CAN baudrate (e.g., 1000), maximum accelerations (e.g., [0.8, 0.8, ..., 0.8]), etc.

The robot description is set up using URDF (Unified Robot Description Format) [51] which is an XML format for representing a robot model.

4 Robot control

4.1 Collision Aware Motion Planning in ROS using a 3D sensor

The aim of collision aware motion planning is to go from a start pose (position and orientation) to a goal pose with no collisions and satisfying all robot constraints. In this regard, sensing is essential to avoid collision when operating in unstructured environments. In particular, it is possible to consider a point cloud which is a set of 3D points that represent the surface of the objects in

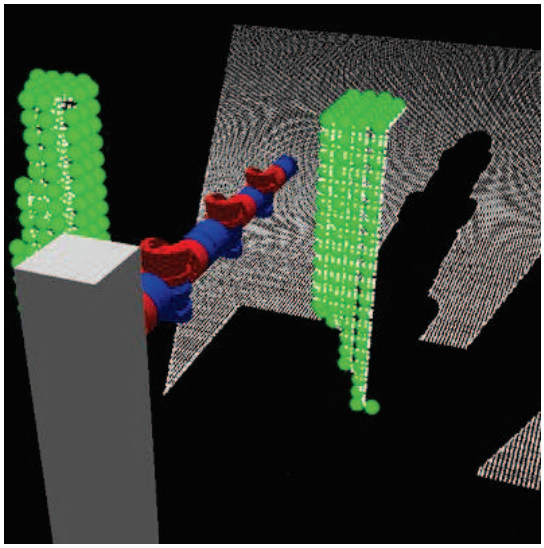

Fig. 6 Point clouds into collision map

the robot environment. The point cloud can be obtained using several types of sensors (e.g., stereo cameras, lasers, Kinect camera, etc.) and can be processed (filtering, segmentation, surface reconstruction, model fitting, etc.) using the Point Cloud Library (PCL) [52]. To perform the planning tasks it is important to filter the 3D points to remove the sensor noise and the robot body (including attached objects).

Environment representation can be divided in two parts: semantic perception and 3D grid representation. Semantic perception represents known objects, i.e., objects recognized using 3D object recognition algorithms, e.g., using the *pcl_recognition* module based on correspondence grouping described in the PCL documentation [52]. This kind of environment representation helps for collision checking and task planning (e.g., pick-up tasks) because it is possible to consider the whole geometry of the object even though it has occluded parts in the point cloud obtained by the sensor. A 3D grid representation is necessary to complete the environment with those parts out of the semantic representation and that must be considered by the planner.

ROS *arm_navigation* stack [53] contains tools for easily generating new robotic manipulation applications using a set of stacks, e.g., the *collision_environment* stack contains tools to create representations of the environment for collision checking. It contains the packages to convert the 3D data obtained from the sensor in the form of point clouds into a collision map using the Octomap library [54] that implements a octree based representation of the environment: probabilistic, flexible and compact 3D mapping, which is optimized for online operation.

The constraint-aware Inverse Kinematics solver (package from *arm_navigation* stack [53]) provides a generic implementation of constraint-

aware kinematics for any serial manipulator. It combines ROS collision checking tools with Orocos KDL [55] forward and inverse kinematics solvers.

OMPL (Open Motion Planning Library) [56] consists of several sampling-based motion planning algorithms. It is basically a repository of planners which allows to choose the most suitable planner for the task and to choose its parameters. Some of the planners available are PRM, RRT, ESTS, SBL, KPIECE, BKPIECE, LBKPIECE, LazyRRT and RRTConnect. The mathematical description and software documentation of all these planners can be found in [56]. ROS interface OMPL through the *ompl_ros_interface* package (*arm_navigation* stack) provides a YAML-based configuration to setup planners. The input to the planner is a ROS motion planning request which specifies the goal as a region in space. It can be a joint space goal (i.e., a nominal joint angle with a tolerance) and a pose goal (i.e., a nominal position and orientation with a tolerance).

Finally, the sampling-based planners can generate jerky trajectories and, hence, smoothing is necessary before sending the planned trajectory to the controller. Cubic spline and maximum velocities/acceleration constraints are used in order to smooth the paths.

4.2 Collision Detection and Force Control

When a robotic device is operating in a human dynamic-changing environment, collisions between the robot and the obstacles cannot always be avoided, e.g., the obstacle might be moving too quick for the robot to react or it might be occluded from the camera field of view. Thus, in order to protect the human operator and the robot system, a proper collision detection method is essential for safe operation. For the Schunk system described in section 3.2, this can be achieved using the force/torque sensor mounted at the robot's end-effector. For this purpose, the force sensor has to be calibrated properly to consider the weight of the different hand or gripper tools, the sensor offsets, as well as the shoe that is being manipulated. The used hand or gripper can be considered as a point mass attached to the end-effector frame. Under this assumption, the force estimation is done as follows:

$$\tilde{\mathbf{f}} = (\mathbf{f} - \mathbf{f_0}) - \mathbf{R}^{\mathbf{F}}_{\mathbf{F}^*}(\mathbf{f}) \cdot \mathbf{f_T} \tag{1}$$

where $\mathbf{f}$ is the vector of measured forces, $\mathbf{f_0}$ the vector of experimentally determined zero offsets of the sensor, $\mathbf{f_T}$ the force vector originating from the tool weight in force sensor coordinates and $\mathbf{R}^{\mathbf{F}}_{\mathbf{F}^*}(\mathbf{f})$ is a matrix describing the rotation between the force sensor frame and the currently measured force vector. Thus, the measurement obtained from the force sensor will be approximately zero if no external forces are applied. Note that the above approach considers forces but not torques.

Therefore, if the robot detects a collision (or the human operator interrupts the actual task), it stops its current task and changes to a force controlled mode, so that the operator is able to move the robot in the Cartesian space by

pulling at the end-effector. Apart from preventing damage to the robot, the operator could for example relieve the system from the collision situation or inspect the carried item. For this purpose, the measured forces are translated into robot motions as follows:

$$\dot{\mathbf{q}}_{\mathbf{ref}} = \mathbf{J}^{\dagger}(\mathbf{q}) \cdot \left(K_p \begin{bmatrix} \mathbf{R}_{\mathbf{B}}^{\mathbf{F}}(\mathbf{q}) \cdot \tilde{\mathbf{f}} \\ \mathbf{0} \end{bmatrix} \right) \tag{2}$$

where $\mathbf{R}_{\mathbf{F}}^{\mathbf{B}}(\mathbf{q})$ is the rotation matrix that aligns the estimated force vector $\tilde{\mathbf{f}}$ with the robot base frame, K_p the proportional gain, $\dot{\mathbf{q}}_{\mathbf{ref}}$ the reference velocities sent to the robot controller and $\mathbf{J}^{\dagger}(\mathbf{q})$ the pseudo-inverse of the current robot Jacobian $\mathbf{J}$. However, since the elements of the pseudo-inverse matrix tend to infinity for singular robot configurations, an adaptive damped least squares solution is considered to limit the joint velocities when the robot is close to a singular configuration. In particular, the Orocos Kinematics and Dynamics Library (KDL [55]) provides the Weighted Damped Least-Squares (WDLS) inverse kinematics solver with an adaptive damping term $k(\mathbf{q})$ as follows:

$$\mathbf{J}^{\dagger}(\mathbf{q}) = \mathbf{J}^{\mathbf{T}} \Big(\mathbf{J}\mathbf{J}^{\mathbf{T}} + k(\mathbf{q})^2 \mathbf{I} \Big)^{-1} \tag{3}$$

where the damping term is computed with respect to the current *manipulability* $M(\mathbf{q})$ of the robot:

$$k(\mathbf{q}) = k_0 \cdot \left(1 - \frac{M(\mathbf{q})}{M_0} \right) \tag{4}$$

with

$$M(\mathbf{q}) = \sqrt{|\mathbf{J}(\mathbf{q}) \cdot \mathbf{J}(\mathbf{q})^{\mathbf{T}}|} \tag{5}$$

where k_0 is the maximum desired damping and M_0 the maximum manipulability, which is experimentally determined. Thus, the damping factor k converges to zero when being far from singular configurations and tends to k_0 when the robot is close to a singularity. Therefore, the system is guided safely through a singularity while following the desired reference with reduced accuracy.

4.3 Object detection and modelling

The object modeling/detection algorithm of the ROS stack of the RoboEarth project [57] has been used. It allows to build up a model of an object (not too big and non-transparent) using the Kinect sensor. Based on this model, the object can be found afterwards in a cluttered scene and the camera relative pose can be determined. In the modeling phase, the object is placed onto two sheets of paper, where Augmented Reality Markers are printed on. The Markers are tracked to reconstruct the camera pose relative the markers. When the camera is moved around the object (or the object is moved in front of the camera), the camera pose is calculated continuously and the SURF features are extracted using OpenCV implementation. After that, the 3D position is

Fig. 7 Object detection applied to a shoe

determined using the depth information from the Kinect sensor. The local point cloud of the object and the SURF features are stored for several frames. These are the reference poses that are used for the object detection. Thus, the SURF features are extracted from the camera image and matched sequentially against the stored feature descriptor for each reference pose of the learned objects. This process can be rather time consuming depending on the number of objects and the number of reference poses. SURF features show a good scaling and rotational invariance in 2D, but not so good for 3D detection. Therefore, many reference poses per object are required for robust 3D object detection. Fig. 7 shows a picture of the shoe detection/matching algorithm.

4.4 Grasp planning with constrained contact regions

This subsection outlines an algorithm to plan grasps using the object representation. The algorithm is based on the grasp hypothesis generation implemented in the Simox toolbox [58]. The algorithm presented there was extended to cope for restricted touch regions on objects. The input for the algorithm are a kinematic model of the robotic hand, a 3D mesh of the object to grasp and a grasp definition. The grasp definition contains a pre-shape of the hand and an approach vector. On the object, a random approaching point x is sampled on which the vector n_x normal to the surface is approximated. Then, the pre-shape is aligned to this normal vector using the pre-shape approach vector. The position in the space is selected in a way that with the hand opened (in pre-shape) there is no initial collision with the object. To do so, the collision checking is performed starting with a position close to the hand and moving the hand along the approach vector in negative direction. When there is no initial collision for a position, the hand is closed with a constant speed until all fingers are in contact. If there are more than two contacts, it is checked if all contacts are within the valid grasp regions. If so, grasp quality is calculated

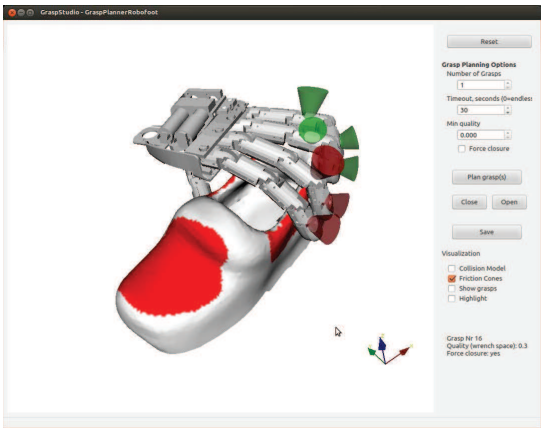

Fig. 8 Grasp planning with restricted touch regions

based on the number of contact points and it is checked if the force closure property of the grasp is fulfilled.

Fig. 8 shows a picture of the grasp planning with red regions. The red areas on the object indicate regions in which the robot is not allowed to touch the object (process manually performed). The green friction cones indicate contact points that are valid, while the red cones indicate invalid contacts. In the case of the displayed grasp, the grasp planning algorithm would reject it.

5 Experimental Results

An essential prerequisite for applying robotics technologies in footwear packaging and/or the automatic manipulation of relevant objects such as shoes or shoe boxes is the ability of the robotic system to automatically detect these objects in the environment. By acquiring 3D sensor data and color information of an object from different viewpoints, object models can be created in order to detect them in the robotic scene.

Literature distinguishes between local and global object representations. On the one hand, local methods [59] extract the so-called feature points from the object views and perform a classification on the regions around these points. On the other hand, global object representations are based on statistical classification techniques where complete object views are used to train an object classifier [60]. Global approaches are powerful in deciding whether or not a detected object that is not included in the training set (e.g., a different perspective is used), belongs to a known object. Whereas, local feature point matchers have advantages in detecting partially visible objects, e.g., in cluttered scenes. However, both methods give limited information about the object accurate position and orientation, which is required for its manipulation in the 3D space. Hence, geometric model-driven approaches are used to fit a 3D shape model into the scene and to track it [61]. These methods reli-

ably compute the pose information under the assumption that there is a priori knowledge of the location where the model fitting is started.

This section presents two experiments on two different platforms to show the effectiveness of the trajectory planning algorithm to avoid the obstacles and the motion algorithm using force control.

5.1 Trajectory planning avoiding obstacles

In order to test the implementation presented in subsection 4.1, several experiments have been performed. The computed motion planning should avoid the collisions between the robot (or the object that is carrying) and the objects of the environment. The objects have been detected using a Kinect camera and a point cloud has been generated to get an image similar to the one presented in Fig. 6. Fig. 9 shows a sequence in which the robot is carrying an aluminum stick from one side to the other of a rigid structure, which is marked with black and yellow bands. The motion planning takes into account the initial and final object's pose, the shape of the transported object and the environment in order to properly avoid collisions.

5.2 Motion generation using force control

Fig. 10 shows an experiment, which has been conducted to show the functionality of collision detection and force control, see section 4.2. The manipulator shall move the grasped shoe from one shoe box to another, both of which are located on different tables. In case an unexpected collision is detected by the force sensor, the robot automatically switches to a force controlled mode, where it can be moved freely by the human operator. If no more external forces are applied, it continuous following its previous trajectory. The modification of a pre-planned trajectory using a reactive force controller without an additional task supervision component might lead the robot to significantly deviate from the desired trajectory to accomplish the given task. To make the execution more robust, the task could be considered as failed at some point and a re-planning component, which eventually makes use of a vision sensor, provides a new trajectory.

5.3 Shoe grasping

Once the object has been successfully detected, a suitable grasping position is computed and the object is grasped. According to Taylor [62] there are three main techniques for gripping non-rigid objects:

- Mechanical surface, in which the material is clamped or pinched between gripper fingers to give high frictional holding.
- Intrusive, in which pins are fed into the surface or body of the material.

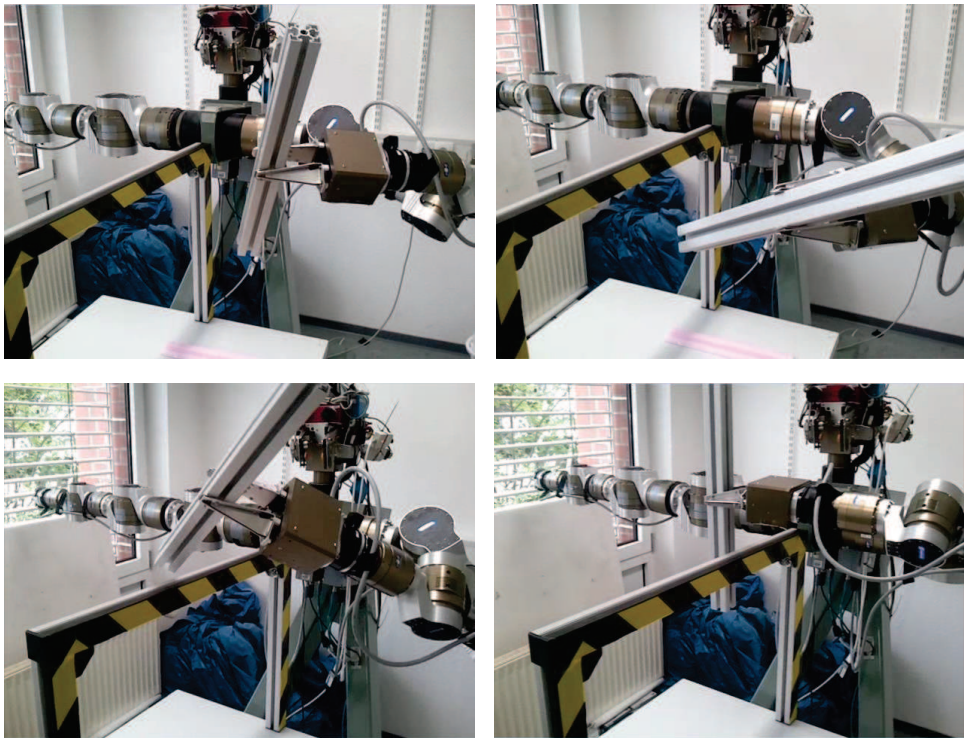

Fig. 9 Automatic trajectory re-planning to avoid obstacle collision (A video of this experiment can be played at https://media.upv.es/player/?id=043d12e0-8135-11e6-8f97-b7415bf6c110)

– Surface attraction, which includes the use of adhesives or vacuum.

Initial tests were performed with industrial parallel grippers at both research laboratories, see Fig. 9 and Fig. 10. However, if certain shoe areas have to be avoided (e.g., not touched, see Fig. 8), the use of a multifinger hand provides a higher search space on which to find a suitable hand configuration that robustly grasps the shoe and, at the same time, avoids touching certain shoe areas.

The grasping algorithm designed in this work is independent of the used hardware platform, i.e., the gripper hand of the robot. In this regard, tests have been performed for the grasping algorithm considering a peer review procedure, i.e., checking that the grasping algorithm was properly executed by an iCub robotic hand (by Istituto Italiano di Tecnologia) [63] and an IH2 Azzurra robotic hand (by Prensilia) [64]. For cases in which other algorithms were checked (e.g., the trajectory planning), a Schunk pneumatic gripper was used.

Fig. 11 shows the process of locating the shoe, performing motion planning, reaching, and, finally, grasping. Using the previously defined detection

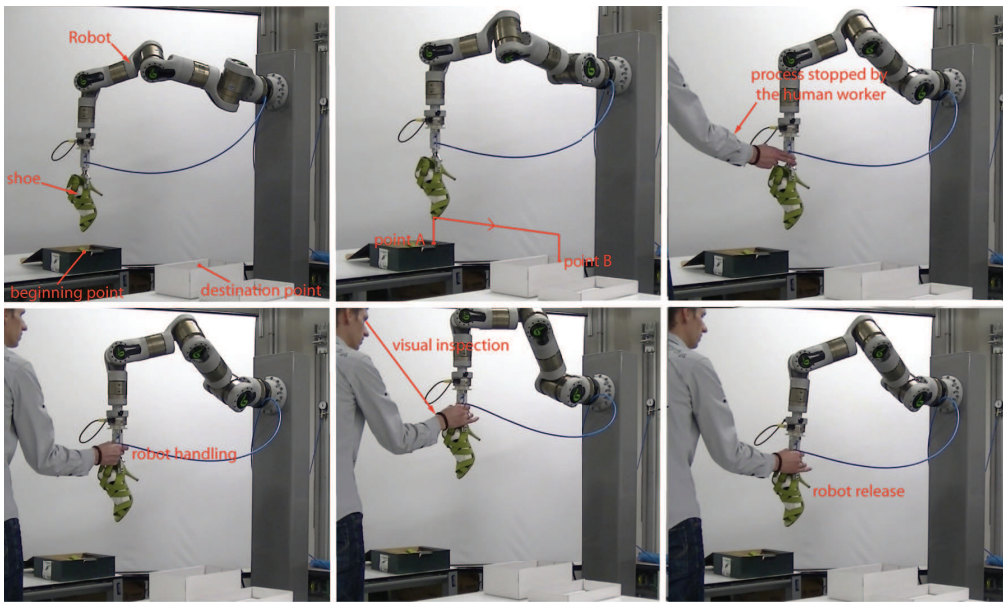

Fig. 10 Experiment on force detection and force control (A video of this experiment can be played at https://media.upv.es/player/?id=2a8aa2b0-8134-11e6-8f97-b7415bf6c110)

methods, the Kinect camera is used to detect the shoe and identify its relative pose to the table. The table is considered an obstacle for the planning algorithm. Given a certain desired grasp area for the shoe, the motion planner is requested to find an optimal trajectory which brings the right arm from its current position to a fixed distance relative to the shoe and with the approaching orientation vector according to the detected pose of the shoe. The final shoe grasping from here will be performed 'blindly' and the robot will move along the approaching vector towards the shoe and close its fingers. Tactile information on the tips and palms of the hand will provide the necessary contact information to complete the grasp motion. In this case, compared to parallel grippers, the use of a multifinger hand provides a higher level of robustness on the closing grasp, as the fingers will enclose the shoe within its fingers and the grasp point location.

6 Key Findings and Remaining Challenges

The key findings achieved in this work are:

- Analysis of the shoe industry as a whole, identifying which activities can be automated and which ones remain still unsolved.
- Automation of the shoe packaging process, testing issues like: robust object detection in complex environments (see Fig. 6), shoe grasping considering a

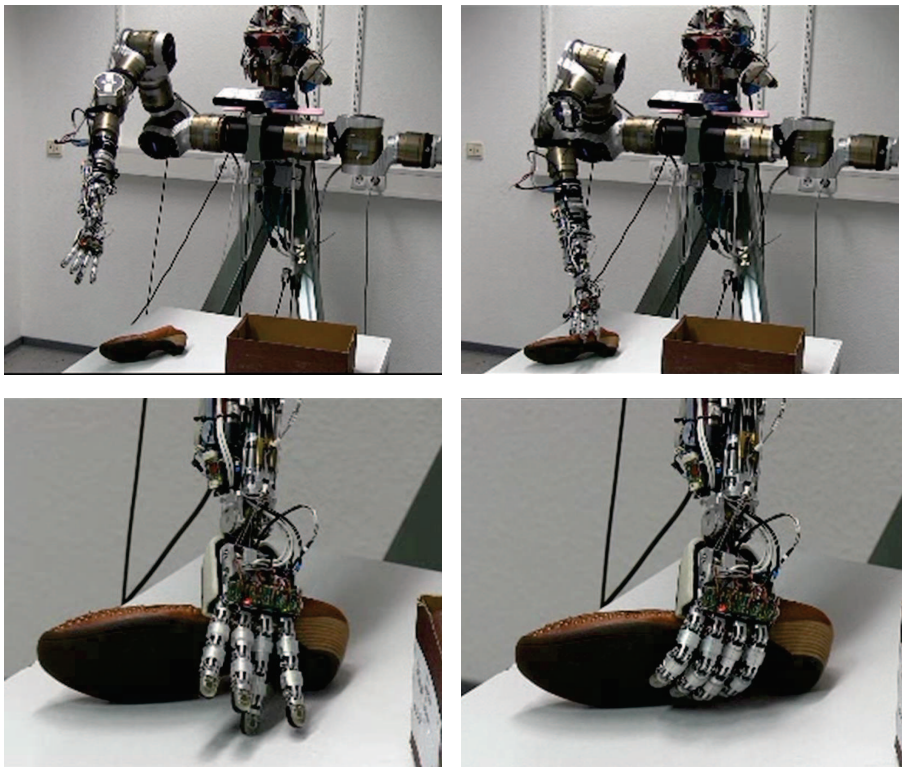

Fig. 11 Complete shoe grasping process including shoe detection, motion planning, reaching and grasping

blind peer review (using two different dexterous hands and getting equivalent results), trajectory planning with collision avoidance, force control for operator interaction, etc.
- Making progress towards the automation of the shoe industry.
- Identification of remaining challenges in shoe automation.

The remaining challenges of this work are:

- Automation of shoe cleaning.
- Automation of shoe polishing.
- Automation of sewing.
- Automation of leather pre-shaping.
- Manipulation of tiny pieces.
- Automation of shoelaces tie.
- Speed-up of robotic movements to increase in productivity.

In order to illustrate how these remaining challenges could be tackled, the first two items are analyzed. Cleaning or polishing is required depending on the shoe model. On the one hand, cleaning is mainly needed in sport and

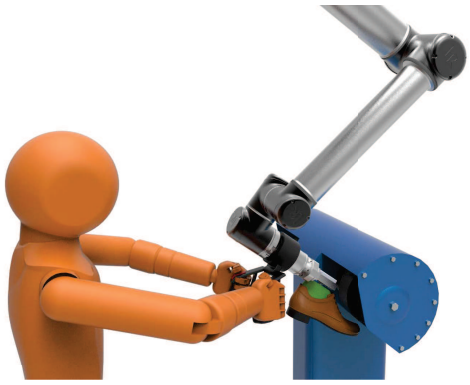

Fig. 12 Operator teaching the robot to clean the shoe.

Fig. 13 Robot performing a cleaning task before introducing the shoe inside the box.

casual shoes and the cleaning areas are usually the same due to the production process. On the other hand, polishing is usually required in classic shoes, where the design has some shaded lines of style that have to be generated by polishing. For instance, the automation of the shoe cleaning or shoe polishing could be accomplished with the following two steps: firstly, the human operator manually moves the robot to perform the task and the robot records the corresponding movements and forces, see Fig. 12; secondly, the robot repeats the task autonomously, as shown in Fig. 13.

7 Conclusion

This paper has shown a detailed description of the industrial shoe packaging process using a robotic system. For this purpose, the technical difficulties of introducing a robotic system in such scenario have been analyzed. Furthermore, the required hardware, software, and special tools have been described and their advantages and disadvantages have been discussed in order to make a proper selection of the components. For instance, some tools like the ROS

arm_navigation stack have been especially helpful to solve the manipulation path planning in the complex environment given by the shoe packaging process. A set of experimental setups have been devised to show the effectiveness of the proposed approach as well as the advantages and disadvantages of the used components and their underlying technology.

Acknowledgements This work has been partly supported by Ministerio de Economia y Competitividad of the Spanish Government (Key No.: 0201603139 of Invest in Spain program and Grant No. RTC-2016-5408-6) and by the Deutscher Akademischer Austauschdienst (DAAD) of the German Government (Projekt-ID 54368155).

References

1. N. Pedrocchi, E. Villagrossi, C. Cenati, and L.M. Tosatti. Design of fuzzy logic controller of industrial robot for roughing the uppers of fashion shoes. *The International Journal of Advanced Manufacturing Technology*, 77(5):939–953, March 2015.
2. J.J. Hinojo-Prez, M. Davia-Aracil, A. Jimeno-Morenilla, L. Snchez-Romero, and F. Salas. Automation of the shoe last grading process according to international sizing systems. *The International Journal of Advanced Manufacturing Technology*, 85(1):455–467, July 2016.
3. J.V. Dura-Gil, A. Ballester-Fernndez, M. Cavallaro, A. Chiodi, A. Ballarino, V. von Arnim C. Brondi, and D. Stellmach. New technologies for customizing products for people with special necessities: project fashion-able. *International Journal of Computer Integrated Manufacturing*, In Press(DOI: 10.1080/0951192X.2016.1145803), 2016.
4. F. Jatta, L. Zanoni, I. Fassi, and S. Negri. A roughing/cementing robotic cell for custom made shoe manufacture. *International Journal of Computer Integrated Manufacturing*, 17(7):645–652, 2004.
5. B. Nemec and L. Zlajpah. Robotic cell for custom finishing operations. *International Journal of Computer Integrated Manufacturing*, 21(1):33–42, January 2008.
6. R. Molfino et al. Modular, reconfigurable prehensor for grasping and handling limp materials in the shoe industry. In *IMS International Forum, Cernobbio*, 2004.
7. Intelishoe - integration and linking of shoe and auxiliary industries. 5Th FP.
8. Special shoes movement. 7th FP, NMP-2008-SME-2-R.229261, http://www.sshoes.eu.
9. J.L. Vilaca and J. Fonseca. A new software application for footwear industry. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pages 1–6, oct. 2007.
10. Custom, environment and comfort made shoe. 6TH FP [2004-2008].
11. Framework of integrated technologies for user centred products. Grant agreement no.: CP-TP 229336-2. NMP2-SE-2009-229336 FIT4U -7TH FP.
12. Robofoot project website. http://www.robofoot.eu/. Accessed 2016/09/16.
13. E. Montiel. Customization in the footwear industry. In *Proceedings of the MIT Congress on Mass customization*, 2007.
14. I. Sucan and L.E. Kavraki. A sampling-based tree planner for systems with complex dynamics. *Robotics, IEEE Transactions on*, 28(1):116–131, feb. 2012.
15. J.J. Jr. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 995–1001 vol.2, 2000.

16. Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 489–494, may 2009.
17. Oliver Brock and Oussama Khatib. Elastic strips: Real-time path modification for mobile manipulation, 1997.
18. T. Kroger. Opening the door to new sensor-based robot applications #x2014;the reflexxes motion libraries. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, may 2011.
19. Jur van den Berg, David Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2366–2371, May 2006.
20. Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A robot path planning framework that learns from experience. In *IEEE International Conference on Robotics and Automation*, pages 3671–3678. IEEE, 2012.
21. Rainer Bischoff, Johannes Kurth, Guenter Schreiber, Ralf Koeppe, Alin Albu-Schaeffer, Alexander Beyer, Oliver Eiberger, Sami Haddadin, Andreas Stemmer, Gerhard Grunwald, and Gerhard Hirzinger. The kuka-dlr lightweight robot arm - a new reference platform for robotics research and manufacturing. *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–8, june 2010.
22. Brian Rooks. The harmonious robot. *Industrial Robot-an International Journal*, 33:125–130, 2006.
23. N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann. Visual servoing for humanoid grasping and manipulation tasks. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 406–412, dec. 2008.
24. R.S. Pieters et al. Direct trajectory generation for vision-based obstacle avoidance. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.
25. Kinect for windows sensor components and specifications, website. http://msdn.microsoft.com/en-us/library/jj131033.aspx. Accessed 2016/09/16.
26. F. Jatta, L. Zanoni, I. Fassi, and S. Negri. A roughing cementing robotic cell for custom made shoe manufacture. *International Journal of Computer Integrated Manufacturing*, 17(7):645–652, 2004.
27. I. Maurtua and A. Ibarguren A. Tellaeche. Robotics for the benefit of footwear industry. In *International Conference on Intelligent Robotics and Applications*, pages 235–244. Springer Berlin Heidelberg, 2012.
28. R.C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.
29. N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, 1980.
30. H. Asada and J.-J.E. Slotine. *Robot Analysis and Control*. Wiley, 1986.
31. ROS official web page. *http://www.ros.org*, (Accessed on 2017/02/03).
32. B. Langmann, K. Hartmann, and O. Loffeld. Depth camera technology comparison and performance evaluation. In *1st International Conference on Pattern Recognition Applications and Methods*, pages 438–444, 2012.
33. The player project. free software tools for robot and sensor applications. *http://playerstage.sourceforge.net/*, (Accessed on 2017/02/03).
34. Yet another robot platform (YARP). *http://www.yarp.it/*, (Accessed on 2017/02/03).
35. The OROCOS project. smarter control in robotics and automation. *http://www.orocos.org/*, (Accessed on 2017/02/03).
36. CARMEN: Robot navigation toolkit. *http://carmen.sourceforge.net/*, (Accessed on 2017/02/03).
37. ORCA: Components for robotics. *http://orca-robotics.sourceforge.net/*, (Accessed on 2017/02/03).
38. MOOS: Mission oriented operating suite. *http://www.robots.ox.ac.uk/ mobile/MOOS/wiki/pmwiki.php/Main/HomePage*, (Accessed on 2017/02/03).
39. Microsoft robotics studio. *https://www.microsoft.com/en-us/download/details.aspx?id=29081*, (Accessed on 2017/02/03).

40. Pr2 ros website. http://www.ros.org/wiki/Robots/PR2. Accessed 2016/09/16.
41. Care-o-bot 3 ros website. http://www.ros.org/wiki/Robots/Care-O-bot. Accessed 2016/09/16.
42. Aila, mobile dual-arm manipulation, website. http://robotik.dfki-bremen.de/de/forschung/robotersysteme/aila.html. Accessed 2016/09/16.
43. Package libpcan documentation, website. http://www.ros.org/wiki/libpcan. Accessed 2016/09/16.
44. Pcan driver for linux, user manual. http://www.peak-system.com. Document version 7.1 (2011-03-21).
45. Pcan driver for linux, user manual. http://wiki.ros.org/schunk_powercube_chain. Accessed 2016/09/16.
46. Ros nodes documentation, website. http://www.ros.org/wiki/Nodes. Accessed 2016/09/16.
47. Ros messages documentation, website. http://www.ros.org/wiki/Messages. Accessed 2016/09/16.
48. Ros topics documentation, website. http://www.ros.org/wiki/Topics. Accessed 2016/09/16.
49. Ros services documentation, website. http://www.ros.org/wiki/Services. Accessed 2016/09/16.
50. Yaml files officials website. http://www.yaml.org/. Accessed 2016/09/16.
51. Ros robot model (urdf) documentation website. http://www.ros.org/wiki/urdf. Accessed 2016/09/16.
52. Point cloud library (pcl), website. http://www.pointclouds.org/. Accessed 2016/09/16.
53. Arm navigation ros stack, website. http://wiki.ros.org/arm_navigation. Accessed 2016/09/16.
54. Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 2013.
55. Orocos kdl documentation, website. http://www.orocos.org/kdl. Accessed 2016/09/16.
56. Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. url-http://ompl.kavrakilab.org.
57. M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J.M.M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft. Roboearth. *Robotics Automation Magazine, IEEE*, 18(2):69–82, 2011.
58. Simox toolbox. http://simox.sourceforge.net/. Accessed 2016/09/16.
59. Pierre Moreels and Pietro Perona. Evaluation of features detectors and descriptors based on 3d objects. *International Journal of Computer Vision*, 73:263–284, 2007.
60. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, 2001.
61. C. Teuliere, E. Marchand, and L. Eck. Using multiple hypothesis in model-based tracking. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4559–4565, may 2010.
62. V. C. Moulianitis, A. J. Dentsoras, and N. A. Aspragathos. A knowledge-based system for the conceptual design of grippers for handling fabrics. *Artif. Intell. Eng. Des. Anal. Manuf.*, 13(1):13–25, January 1999.
63. S. Davis, N.G. Tsagarakis, and D.G. Caldwell. The initial design and manufacturing process of a low cost hand for the robot icub. In *8th IEEE-RAS International Conference on Humanoid Robots*, pages 40–45, 2008.
64. G. Cerruti, D. Chablat, D. Gouaillier, and S. Sakka. Design method for an anthropomorphic hand able to gesture and grasp. In *IEEE International Conference on Robotics and Automation*, pages 3671–3678. IEEE, 2015.