A Flexible and Collaborative Approach to Robotic Box-Filling and Item Sorting

Pietro Balatti[a,b,*], Mattia Leonori[a] and Arash Ajoudani[a]

[a]HRI[2] Lab, Istituto Italiano di Tecnologia, via San Quirico 19D, 16163 Genoa, Italy
[b]Dept. of Information Engineering, University of Pisa, Pisa, Italy

ARTICLE INFO

Keywords:
Intelligent and Flexible Manufacturing
Human-Robot Collaboration
Autonomous Agents
Manipulation Planning

ABSTRACT

In this paper, we introduce an adaptive robotic manipulation framework to respond to the flexibility needs of common industrial tasks such as box-filling and item sorting. The proposed framework consists of a vision module and a robot control module. The vision module is responsible for the detection and tracking of the environment (e.g., box and the items), which is also capable of creating an occupancy grid in real-time, to continuously update the robot trajectory planner with the occupied portions of the detected box and their coordinates. The robot control module includes a trajectory planner and a self-tuning Cartesian impedance controller, to implement an adaptive strategy for the picking, placement, and sorting of the items in the box. The item-sorting strategy is based on our preliminary observations on human motor behavior, implementing a trade-off between the task execution accuracy and environmental perception uncertainty. The efficacy of the framework in performing a flexible box-filling task using a robot, autonomously or in collaboration with a human, is evaluated through several experiments.

1. Introduction

Recent collaborative robotic technologies have the potential to add high levels of flexibility to the manufacturing processes, due to their versatility and human-centric design [1]. They can not only contribute to the creation of hybrid (human-plus-robot) and resource-efficient manufacturing solutions, but also can help reduce human physical stress and automate repetitive and cognitively unexciting industrial tasks [2]. Despite this, cobots of today are mostly exploited in structured manufacturing environments, where a precise knowledge of the surroundings is required for their operation. In such a way, the true potential of cobots cannot be demonstrated, which is fast adaptation to the variabilities arising from the environment and human co-workers.

Several research works have aimed to improve the manipulation flexibility of cobots in performing repetitive tasks such as pick-and-place [3, 4], sorting [5, 6], and boxing [7], through adaptive control systems [8, 9, 10] that exploit sensory information such as vision [11, 12, 13], and force [14, 15]. These operations are particularly important from a flexible automation perspective, since they are of a repetitive nature, and involve large body movements that can lead to musculoskeletal disorders in long term [16, 17].

The authors in [18] propose a control, where an online estimation of the object pose is adopted, based on force sensing during interaction and vision information provided by a camera. Although the object pose is time-varying, its geometry is modeled and known a priori. In [19], a multi-modal sensor-based control framework for human-robot collaboration is proposed. The robot is able to adapt to the human intention, modifying its control strategy accordingly.

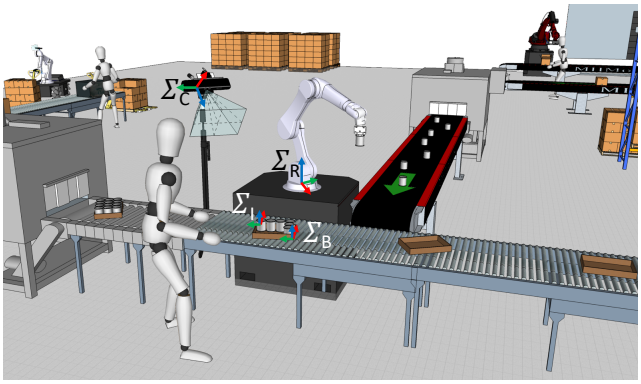

Figure 1: Concept illustration of flexible and collaborative robotic box-filling and item-sorting in manufacturing industry. The depicted reference frames represent the robot base (R), the camera (C), the box (B), and the item (I).

Several bin packing strategies, addressing the problem of item sorting, have been introduced over the years in the field of operational research [20]. Although both exact algorithms [21, 22] and heuristics [23, 24, 25, 26] have been considered, they mainly rely on a priori knowledge of the items order, besides being computationally expensive. Furthermore, these studies do not consider the variations brought by the human counterparts (e.g., changes in item sorting) and are mainly evaluated and performed in simulation studies. Hence, they could find their way into practice only with a perfect knowledge of the surrounding environment and an accurate robot trajectory planning, which are mainly typical of machine tools that do not allow for collaborative approaches.

To the best of the authors' knowledge, very few appli-

This work was supported by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 871237 (SOPHIA).

*Corresponding author
pietro.balatti@iit.it (P. Balatti)
ORCID(s): 0000-0001-8303-9733 (P. Balatti)

cations of automated box-filling and item sorting solutions have been developed. Sgarbossa et al. [27] presented a robot picker that can work alongside manual order picking, developing a method to assign products to different warehouse zones. Boudella et al. [28] developed a hybrid kitting system that is composed of a robot and an operator working in series with the aim of delivering parts to a Just-In-Time mixed-model assembly line. However, these existing solutions are mainly devoted to allocating roles to robots and their human counterparts, therefore tackling the item sorting problem from a higher decision-making level, and not in terms of physical sorting of articles inside a container.

To respond to the flexibility needs of common industrial tasks such as pick-and-place and item-sorting, in this work, we propose a new adaptive framework for collaborative robots. The aim is to provide the system with the ability to cope with unexpected environmental and operational changes (e.g., human interventions and box position/orientation changes). To this end, we first decided to perform human experiments to understand how humans perform typical pick-and-place and item-sorting tasks subject to perception uncertainties. Starting from this analysis, we defined a novel human-inspired *items placing strategy* that has been embedded in the presented framework. This strategy allows both for picking objects from a conveyor and placing them in a box, and for reorganizing items that are already lying within the container, so as to compact them to create more room for other articles.

The first component of this framework is a vision module responsible of the detection and tracking of the items and the box, where the items will be sorted. The module enables an arbitrary placement of the box in robot workspace, which can be even subject to displacements during the execution of the task. The vision module is also capable of creating an occupancy grid in real-time, to update the robot trajectory planner with the occupied/free portions of the detected box and their coordinates. The update is triggered at every iteration of the task, i.e. every time a new item has to be sorted in the box. Additionally, it can also detect items centers, to allow a reorganization of the items within the box.

Robot control module consists of a trajectory planner, which implements an adaptive strategy for the placement and sorting of the items in the box. The execution of the planned trajectories is achieved by a self-tuning Cartesian impedance controller, which facilitates the adaptation of the robot end-effector trajectories to the environmental constraints along its path. This is achieved by implementing a stiff behavior along the planned trajectory, and a compliant profile in other Cartesian axes, which are tuned in real-time based on the vision and interaction force information.

To evaluate the adaptation capacity of the proposed framework to the variations of the task or the environment, we designed a collaborative box-filling and item-sorting experiments. In our setup, a box is arbitrarily placed in front of a cobot, whose task was to pick, place, and sort a number of motor shells in it. We demonstrate that the proposed framework is robust to changes in box location, and also to human interventions where an item is added or removed from the box during the execution of the task. This enables a human worker to intuitively collaborate with the cobot to fill in the box with the target items, to perform it faster (similar to a two-person operation), otherwise, the robot is capable of performing the task autonomously. The experimental results additionally illustrate the advantage of the developed self-tuning controller in comparison to a stiff or a compliant interaction behavior. Finally, we show how the framework is capable of reorganizing items already present in the box. In fact, if they have been randomly placed, for instance due to an operator not compacting them, this new method allows for a more efficient packing.

2. Observations of human motor behavior

To implement a practical *items placing strategy* for the robotic box-filling, we decided to get inspiration from human motor behavior, asking 15 human subjects to fill in some items in a box in the most natural way possible. The objective was to understand the underlying perception-action coordination strategies for this particular task, and to possibly replicate them in our robot control framework. The subjects involved in the experiments were not professional packers, since the goal here was not to understand the best way humans do packing, but rather to learn how uncertainties in perception affect packing actions.

In the depicted scenario, the robotic end-effector can be regarded as the human hand, and the camera perception system as the human eyesight. However, we know that steerable human sight is much more accurate w.r.t. the artificial perception sensing of a fixed camera, especially in unstructured and dynamically changing environments. Hence, we decided to carry out the experiment twice for each subject: once with the eyes uncovered to simulate a perfect visual perception system, and once with the subjects being blindfolded to reproduce a scenario with complete lack of visual sensing.

The subjects were asked to place one item in the lower-right edge of a box (see "goal" reference frame $\mathbf{\Sigma}_G$ in Fig. 2a), while their hand pose ($\mathbf{\Sigma}_H$) was tracked with a *Intel RealSense Depth* camera through the OpenPose library [29]. Conventionally, in industrial pick-and-place machine tools and robotic systems, the items are directly placed in the desired pose, relying on accurate sensing of the environment and precise robot trajectory planning. Therefore, the main objective of this analysis was to infer how the items were placed in the goal position that was instructed to the subjects (the box edge).

To retrieve the human items placing strategy, we decided to analyze the trajectory executed from the moment the item was in contact with the box layer till it was released in its goal pose. Fig. 2a shows the three different poses of the human arm before reaching the contact with the box (P1), during the box layer contact (P2), and in the ending pose where the item had to be placed (P3). Fig. 2b depicts the human pose w.r.t. the Box frame Σ_B. From here it is possible to notice

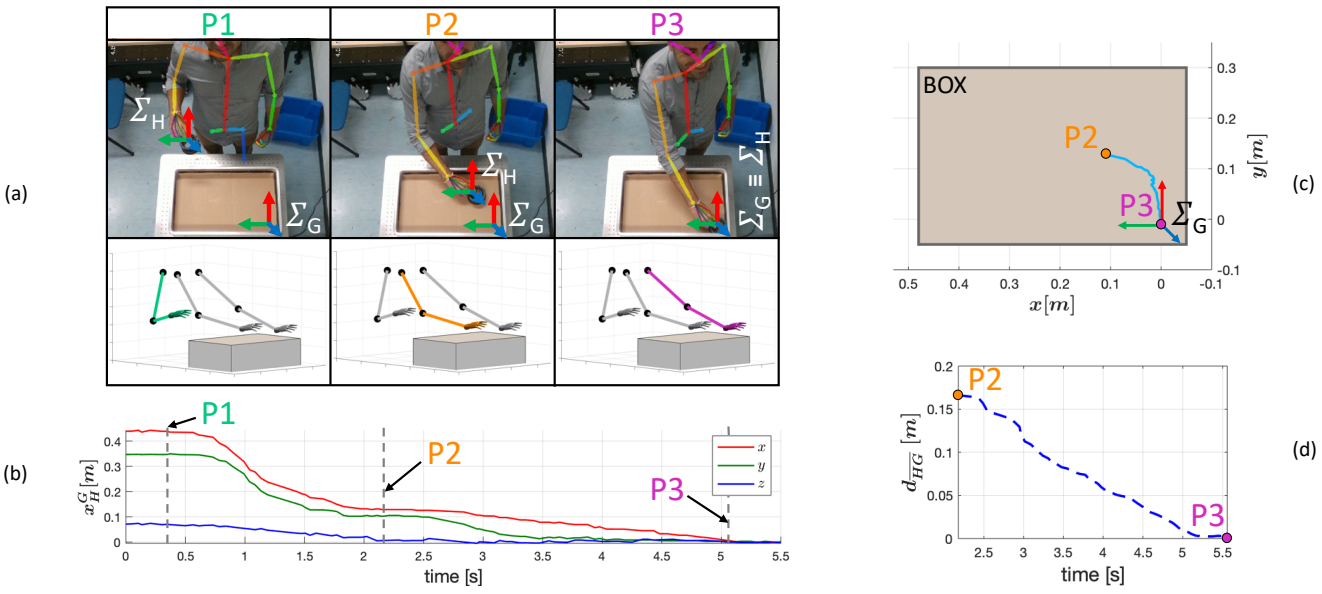

Figure 2: Three different poses of the human arm have been illustrated in this figure: before reaching a contact with the box (P1), during the contact (P2), and the pose where the item had to be placed (P3). On the left side, subfigure (a) depicts the experimental snapshots of the three poses and their graphical representations, while (b) shows the human hand pose $\boldsymbol{X}_H^G$ w.r.t. the goal reference frame $\boldsymbol{\Sigma}_G$: when $\boldsymbol{X}_H^G(z)$ tends to a null value we can consider the start of the sliding phase on the plane (P2). On the right side, the plot of the sliding phase (P2 to P3) is represented in the xy plane (c), as well as the distance between the hand and the goal $\boldsymbol{d}_{\overline{HG}}$ over time (d).

that, after having reached pose P2, the pose of the human hand on the $z-axis$ (blue curve) is always null, so we can deduce that between P2 and P3 there is only a sliding motion on the horizontal plane xy.

Based on the former considerations, we can illustrate the trajectory computed by the human hand in 2D since the contact with the box layer, i.e. from P2 to P3. Fig. 2c shows the same trajectory tracked in Fig. 2a-b on the box plane, while Fig. 2d shows the Euclidean distance between the hand pose and the goal pose, $\boldsymbol{d}_{\overline{HG}}$, over time. It is worth to notice that the traditional positioning of the item directly on the goal pose (classical *Peg-in-hole* strategy) would make P2 coincide with P3, since the goal pose would be reached simultaneously to the contact with the box plane. With such a strategy, in Fig. 2d, the curve would be degenerated in a single point at the origin of the axes, since a null distance $\boldsymbol{d}_{\overline{HG}}$ would be achieved directly.

Fifteen healthy subjects (twelve males and three females; age, 28.5 $\pm$ 3.9 years; weight, 74.6 $\pm$ 11.4 kg; and height, 178.2 $\pm$ 6.7 cm) participated in the overall experiments. Fig. 3 shows the results of the 15 trials for both the uncovered eyes case and the blindfolded one. Similarly as in Fig. 2c, we illustrate all the subject trials in the box plane (top), and the mean σ and standard deviation μ of $\boldsymbol{d}_{\overline{HG}}$ (bottom).

As can be seen from the results reported in Fig. 3, no subject placed the item directly from the top to the desired pose (*Peg-in-hole* strategy). Instead, although in the first scenario the subjects were coadiuvated by a perfect visual sensing (Uncovered eyes), the strategy employed was to first place the item in an empty area of the box, with an initial distance $\boldsymbol{d}_{\overline{HG}} = 0.05 \pm 0.08\,m$, and then slide it towards the goal

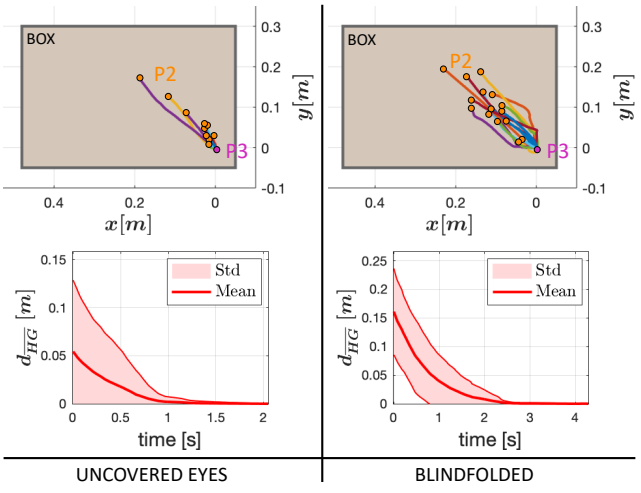

Figure 3: Fifteen subjects performed the item placing experiment: once with uncovered eyes, and once blindfolded. With poorer visual information, the employed sliding strategy is even more evident.

pose. We assume that the reason of this choice stands in the intrinsic confidence of possessing a higher accuracy in sensing the external forces w.r.t. our visual perception. This becomes even more evident in the blindfolded scenario, where the awareness of the environment is very poor because of the lack of eyesight feedback, when $\boldsymbol{d}_{\overline{HG}}$ at time $t = 0$ reaches values three times as high as the first case ($0.16 \pm 0.08\,m$).

The results of this analysis show that the subjects barely chose to place items directly from the top to the desired pose (*Peg-in-hole* strategy), while instead they placed objects start-

ing from a 'safer' initial guess and then adjusted their pose with the help of force feedback. This distance, in our opinion, was decided based on the level of perception uncertainty and the required accuracy. In fact, studies in neuroscience found out that the best performances in reaching tasks are achieved when humans can control both sensory and motor accuracy at the same time, and therefore overcoming visuomotor-only uncertainties [30]. Furthermore, studies on Fitts tapping task [31], simulated through virtual reality, have demonstrated that eye-hand coordination achieves higher performances when enhanced with force feedback [32]. Other results also revealed than blind people possess enhanced tactile acuity, that they developed to overcome the lack of visual feedback [33]. This can also suggest why in our experiment, when blindfolded, subjects relied much more on the force sensing.

Based on these results, we decided to implement an *Items placing strategy* for our robotic system, that will be presented in the next Section. In summary, the algorithm will replicate human behavior in placement of the items in their desired locations (decided by an occupancy grid map and a planner), where a safe initial distance will be chosen experimentally based on the the level of uncertainty present in the perception-action system.

3. Robot framework

The presented framework aims to improve adaptability in the typical industrial task of pick and place. The concept of flexibility is at the core of this methodology, since the environment can be continuously subject to changes induced by the external agents, that can be identified by humans, other robots, or a combination of both.

The framework's software architecture has been built with a high modularity, to facilitate its understanding, and to encourage the reusability of its components. To this end, five main modules have been built, as shown in Fig. 4: (1) a perception module able to detect the location of the items to be picked from the conveyor, the pose of the box where they need to be placed, and the position of the items already lying inside the box, (2) an items placing strategy to place the items inside the box based on the vision information provided by the previous module, (3) an adaptive Cartesian impedance controller with online tunable parameters, (4) a trajectory planner to define the robot desired motions, and (5) a Finite State Machine that coordinates the data exchanged across the other modules.

3.1. Box and items detection

During close-proximity motions, each agent (e.g. humans, robots, automated machines, etc.) demands a considerable level of flexibility and adaptability. For this reason, a reliable and accurate perception system is crucial to perform successful and harmless tasks. The vision components implemented and integrated on the framework are presented, and they are in charge of the conveyor localization, detection of the container and identification of the occupancy of the items lying inside it.

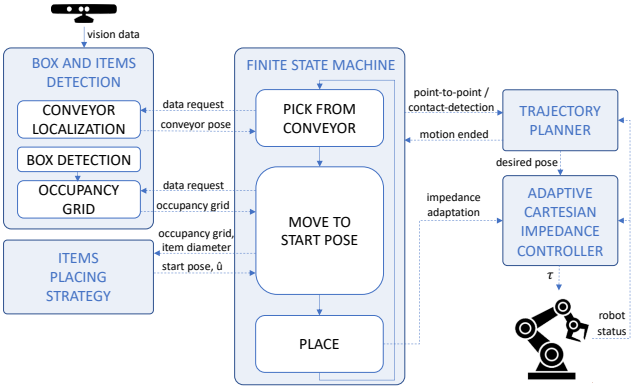

Figure 4: The software architecture of the framework composed of 5 modules implemented as ROS nodes. The dashed lines represent the data exchanged among the units. When a reorganization of the items is triggered, the "Pick from conveyor" state is substituted with a similar primitive to pick the item from the box itself.

Conveyor Localization: the first phase of every pick and place task consists in picking the items from a certain location, in our case represented by a moving conveyor. To augment the framework flexibility, we decided to detect the picking location at every iteration, localizing the conveyor through vision. Since the grasping phase should be performed with a fair precision, we used *ArUco* markers to locate the picking pose of the items on the conveyor. The strength of this approach is the reliability and speed in the marker pose computation, as presented in [34] and [35]. The *ASUS Xtion PRO Live* camera, mounted on top on the table, provides RGB images to the *aruco_detect* ROS package[1], which estimate the pose of the marker. Therefore, a fixed transformation is applied to grasp the object properly.

Box Detection: the second perception requirement consists in the detection and the pose estimation of the box to be filled in. Since ideally the container will be transported throughout the production process and then shipped to other logistic departments, we decided not to use the *ArUco* markers concept applied above. As a matter of fact, every rectangular box is characterized by four corners whose location lies on the same plane. Therefore, we implemented a detection algorithm exploiting the geometric shape of the box. The system processes the point cloud, acquired by the depth sensor integrated in the *ASUS* camera, by applying a pass-through filter. The aim is to remove the points which belong outside the region of interest, defined as $d \leq d_{threshold}$, where $d_{threshold}$ is the parameter which defines the distance between the camera and the table. Through a segmentation procedure, the pre-processed point cloud is clustered in smaller sets of points. Furthermore, the algorithm looks for a set of points that respects specific geometrical features. Thus, by calculating the center of the object and analyzing the principal components, we obtained the reference frame of the object and, accordingly to this, we estimated four candidate corners of the box, based on the dimensions of the

[1] http://wiki.ros.org/aruco_detect

elaborated point cloud. A positive detection is acquired when at least a point is in proximity of each candidate corner.

Occupancy grid: in order to obtain the desired item placement position, the framework evaluates a sorting strategy based on an occupancy grid of the detected box content. The high mutability of the scenario, introduced by the human agent which cooperate with the cobot to accomplish the task, requires a dynamic update of the candidate placement positions. The framework should be capable to neglect non-static objects captured during the detection phase (human upper limb). Therefore, it is necessary to obtain different camera depth frames and merge the information to obtain a single occupancy grid.

To this end, the algorithm processes the point clouds using a pass-through filter, which exploits the pose and the dimensions resulting from the box detection. The bottom of the box is removed from the resulting point clouds, in order to consider just the border of the box and its content. Then, since the occupancy grid is a discretized 2D representation of the space, a projection of the point cloud on the xy plane is required to distinguish the occupied cells from the free ones. In [36], the authors proposed an approach to iteratively update the cell occupancy state. The following formula is based on binary Bayes filter in *log odds* form with an inverse measurement model:

$$l_t^i = l_{t-1} + \log \frac{p(x_i|z_t)}{1 - p(x_i|z_t)} - \log \frac{p(x_i)}{1 - p(x_i)}, \tag{1}$$

where l_t^i is the *log odds* at time t of the i-th cell, x_i is the occupancy state of the cell, $p(x_i)$ is the probability that the i-th cell is occupied, while $p(x_i|z_t)$ is the probability that the i-th cell is occupied given the depth sensor data z at time t. Since, the i-th cell have the same probability to occupied rather than free, we assumed $p(x_i) = 0.5$. In this way (1) can be written as:

$$l_t^i = l_{t-1} + \log \frac{p(x_i|z_t)}{1 - p(x_i|z_t)}. \tag{2}$$

By calculating the *belief* factor, function of *log odds ratio*, the algorithm assigned a value from 0 (free cell) to 1 (occupied cell) to each cell, representing the occupancy probability:

$$bel_t(x_i) = 1 - \frac{1}{1 - \exp l_t^i}. \tag{3}$$

Items detection: usually, in industrial scenarios, components machined and placed in boxes are characterized by the same shape. For this reason, the problem of recognizing the contents of the box has been addressed through a two-dimensional geometric approach. Therefore, an a priori knowledge of the shapes of all the objects that will be arranged in the box is necessary. At the moment, the framework is able to recognize circular items of different radius (e.g. gears, motor shells, etc.), however, the software architecture allows simple integration of different shape detectors. The images acquired by the camera mounted on top of the box are processed through the OpenCV library. Following a first pre-processing through Canny Edge detection, the result obtained is used as input for the Hough Circle detector algorithm, based on Hough Gradient Method [37]. The items thus identified in the two-dimensional image are mapped in 3D and projected in the occupancy grid map.

3.2. Items placing strategy

To ensure flexibility during the placing phase, we assume that at every iteration the items inside the box can be subject to pose changes, for instance due to the collaboration with human agents performing the task in the same container. Mainly, the environmental variables in these situations are identified by two elements: the change of the items quantity/position inside the box and the variation of the box pose in the environment.

In order to correctly place the picked items in a box, we designed a strategy that ensures robustness to potential environmental changes between the placing of an item and its successor, based on the observations (Sec. 2) on humans performing a similar task. The human-inspired strategy includes an initial placing of the object in a collision-free area where no object/border is detected, and then a sliding motion on the box plane towards the desired goal pose identified by the algorithm.

Alg. 1 shows the pseudo-code implementation of the method hereafter described. The open source code implementing this strategy is also available[2]. The algorithm receives as inputs an Occupancy Grid ($\boldsymbol{OG}$), and the diameter of the item $\boldsymbol{d}$, and returns the starting pose $\boldsymbol{start}$ where the item needs to be placed in the box, before being pushed in the direction given by the other output variable, the unit vector $\hat{\boldsymbol{u}}$. This strategy is needed to compact the items inside the box, and augments the tolerance to perception accuracy.

Although, as a proof of concept, we consider the shape of the items to be cylindrical, another item shape would not affect the algorithm core, e.g., for a square-shaped object, we can consider its side as $\boldsymbol{d}$. On the other hand, for non-convex items, the algorithm considers the smallest square with side $\boldsymbol{d}$ that can contain the non-convex item.

The $\boldsymbol{OG}$ information, besides the occupancy probability (*OP*) of each cell, include the *height* of the grid that can be regarded as the grid $\boldsymbol{rows}$, its *width* corresponding to the number of $\boldsymbol{columns}$, its *resolution*, and its *frame* calculated w.r.t. the camera frame. We define $\boldsymbol{D}$ as the minimum number of grid cells where $\boldsymbol{d}$ can fit, by finding the smallest integer greater than or equal to $\boldsymbol{d}$ divided by the grid *resolution*.

The essence of the algorithm resides in the search of an $\boldsymbol{OG}$ subgrid, named *goalGrid*, of dimension $\boldsymbol{D} \times \boldsymbol{D}$ with all empty cells, i.e. with *OP* less than 0.2, and in finding another empty subgrid, named *borderGrid*, of dimension $\boldsymbol{D} \times \boldsymbol{D}$ along the contour of the first one. Like that, the item can be first placed in the *borderGrid* and then pushed towards the *goalGrid* until a contact with the box border or with other already placed items is found. To this end, the algorithm scans all the possible *goalGrids* starting from the one localized at the bottom-right, which corresponds to the $\boldsymbol{OG}$ element

[2] https://gitlab.iit.it/pietrobalatti/items_placing_strategy

Algorithm 1 Items placing algorithm

Input: OG, d
Output: $start, \hat{u}$
  $rows = OG_{height}$
  $columns = OG_{width}$
  $D = ceil(d/OG_{resolution})$
  for $r = 0$ to $rows$-D **do**
    for $c = 0$ to $columns$-D **do**
      $goalGrid \rightarrow (r : r+D, c : c+D)$
      if *emptyGrid(goalGrid)* **then**
        $goal_x = (c + D/2) \cdot OG_{resolution}$
        $goal_y = (r + D/2) \cdot OG_{resolution}$
        $borderGrids = getBorderGrids(goalGrid)$
        for each bg in *borderGrids* **do**
          if *emptyGrid(bg)* **then**
            $bg_{weight} = getOuterCellsOP(bg)$
            candidateStartGrids.add(bg)
          end if
        end for
        if *size(candidateStartGrids)* > 0 **then**
          $startGrid = MinWGrid(candidateStartGrids)$
        else
          $startGrid = goalGrid$
        end if
        $start_x = (startGrid_x + D/2) \cdot OG_{resolution}$
        $start_y = (startGrid_y + D/2) \cdot OG_{resolution}$
        $\vec{u} = goal - start$
        $\hat{u} = \frac{\vec{u}}{|\vec{u}|}$
        return $start, \hat{u}$
      end if
    end for
  end for

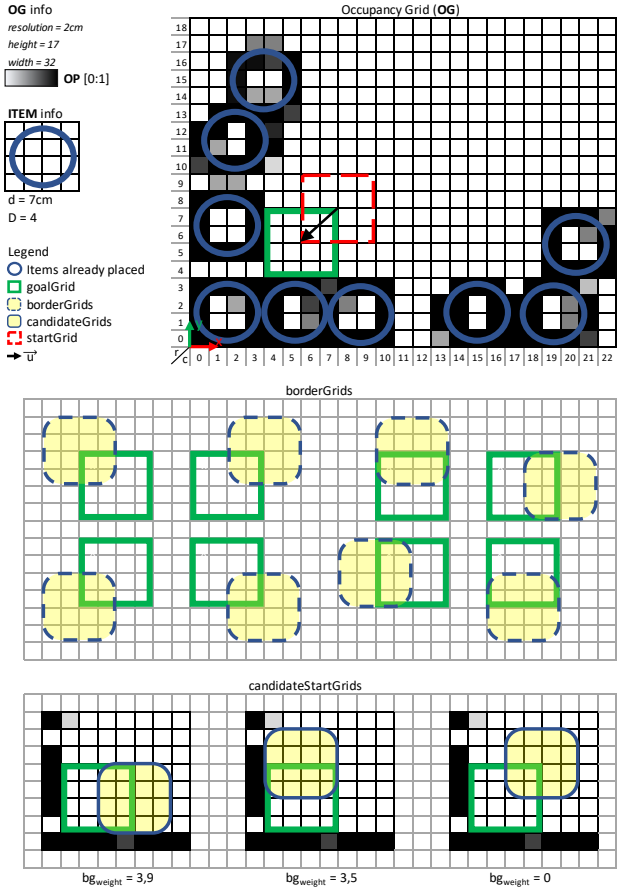

Figure 5: Occupancy grid example (top) with the illustration of the potential *borderGrids* (center), and the *candidateStartGrids* (bottom) that would be extracted with the occupancy grid above.

(0,0), assigning at each iteration a new potential *goalGrid* and checking if it is empty. Once it is found, the ***goal*** pose coordinates *(x,y)* are set as the central element of the examined subgrid multiplied by the ***OG*** *resolution*. The algorithm moves forward to look for the best ***start*** pose coordinates by looking for the *borderGrids*, that can be regarded as the 8 subgrids around the *goalGrid* (see the central part of Fig. 5). Each *borderGrid* that is found to be empty is added to the *candidateStartGrids* list, with associated a *weight* that is the result of the sum of the *OP* of each cell lying on the external outline of the subgrid. Next, the *startGrid* is selected as the one with minimum *weight* among the *candidateStartGrids* list. If the *candidateStartGrids* list is empty, it means that only one ***OG*** subgrid is empty, so the *startGrid* is set equal to the *goalGrid*, implying a peg-in-hole like motion. Similarly as explained above for the *goalGrid*, the ***start*** pose coordinates *(x,y)* are set as the central element of the examined subgrid multiplied by the ***OG*** *resolution*. Finally, the unit vector $\hat{u}$ can be computed normalizing $\vec{u}$, that is the vector connecting ***start*** and ***goal*** coordinates.

To improve the framework efficiency and show enhanced robustness, we also propose an additional strategy aimed to reorganize items already present in the box. The main goal of this strategy is to compact articles lying in the container, to create more space for other items. The core of this method relies on the one presented above in Alg. 1, with a few changes that are described hereafter. First of all, the *Items reorganizing algorithm* receives as input also the items' centers thanks to the *Items detection* module described in Sec. 3.1. Each of these items is checked, so as to retrieve if it has already been placed by the algorithm. If so, the cells of the ***OG*** relative to that item are added to ***OG****, a support occupancy grid that considers only the items already stacked by the strategy (initialized as a copy of ***OG***, without all the portions of the grid containing the detected items). On the other hand, if the item has still to be placed, Alg. 1 is executed taking into account the modified ***OG****. Once the item is placed, its final pose gets added to a vector that stores which items have already been reorganized in the box. Alg. 2 presents a pseudo-code of the illustrated method.

3.3. Adaptive Cartesian impedance controller

During the last decades, most robotized industrial pick and place tasks have been designed with position controlled robots, since they were dealing with structured environments, where all the information was known a priori. However, with

Algorithm 2 Items reorganizing algorithm

Input: $OG, d, detectedItems$
  $OG^* = OG$
  removeOGPortions(OG^*,$detectedItems$)
  for i in $detectedItems$ **do**
    if i in $reorganizedItems$ **then**
      addOGPortion(OG^*,i)
    else
      $start, \hat{u} =$ *Algorithm 1*(OG^*,d)
      $reorganizedItems$.*add*(i)
    end if
  end for

the recent introduction of collaborative robots and in situations that include humans collaborating with robots, these kind of controllers are not anymore suitable, since they lack safety and adaptation capacity. On the other hand, Cartesian impedance control techniques have demonstrated to be able to safely end efficiently achieve any arbitrary quasi-static behavior at the robot end-effector [38, 39].

The robot controller here implemented is based on a Cartesian impedance controller, which can online adapt its impedance parameters. Relying on torque sensing/actuation, the controller is responsible of computing the robot joint torques vector $\boldsymbol{\tau} \in \mathbb{R}^n$ needed to reach the desired poses $\boldsymbol{X}_d \in \mathbb{R}^6$ externally commanded, e.g. by means of a trajectory planner, as:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_{ext}, \tag{4}$$

$$\boldsymbol{\tau}_{ext} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}_c + \boldsymbol{\tau}_{st}, \tag{5}$$

where n is the number of joints, $\boldsymbol{q} \in \mathbb{R}^n$ is the joint angles vector, $\boldsymbol{J} \in \mathbb{R}^{6\times n}$ is the robot arm Jacobian matrix, $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is the mass matrix, $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal matrix, $\boldsymbol{g} \in \mathbb{R}^n$ is the gravity vector and $\boldsymbol{\tau}_{ext}$ is the external torque vector. $\boldsymbol{F}_c$ represents the forces vector in the Cartesian space and $\boldsymbol{\tau}_{st}$ the second task torques projected onto the null-space of $\boldsymbol{J}$. The second task torques are added to keep a configuration as similar as the initial one to avoid joint limit configurations and are given by the difference between the initial joint configuration $\boldsymbol{q}_i(n)$ and the measured joint position $\boldsymbol{q}_m(n)$ pre-multiplied by the difference between the identity matrix $\boldsymbol{I}(n \times n)$ and the product between the Jacobian pseudo-inverse $\boldsymbol{J}^\dagger$ with $\boldsymbol{J}$:

$$\boldsymbol{\tau}_{st} = (\boldsymbol{I} - \boldsymbol{J}^\dagger \boldsymbol{J}) \cdot (\boldsymbol{q}_i - \boldsymbol{q}_m). \tag{6}$$

Forces $\boldsymbol{F}_c \in \mathbb{R}^6$ are calculated as follows:

$$\boldsymbol{F}_c = \boldsymbol{K}_c(\boldsymbol{X}_d - \boldsymbol{X}_m) + \boldsymbol{D}_c(\dot{\boldsymbol{X}}_d - \dot{\boldsymbol{X}}_m), \tag{7}$$

where $\boldsymbol{K}_c \in \mathbb{R}^{6\times6}$ and $\boldsymbol{D}_c \in \mathbb{R}^{6\times6}$ represent respectively the Cartesian stiffness and damping matrix, $\boldsymbol{X}_d$ and $\boldsymbol{X}_m \in \mathbb{R}^6$ the Cartesian desired and measured position, $\dot{\boldsymbol{X}}_d$ and $\dot{\boldsymbol{X}}_m \in \mathbb{R}^6$ their corresponding velocity profiles.

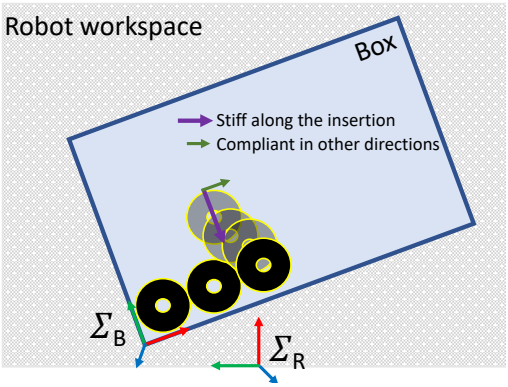

Figure 6: The adaptive impedance controller implements a stiff profile along the motion vector, and a soft profile in other directions, to allow adaptation to the constraints along robot path.

To achieve better interaction performances, the impedance parameters are tuned online so as to keep a compliant profile during most of the motions, unless required by the task. This method has already been demonstrated to guarantee safer interactions when coming in contact with unexpected human collisions [40], and unknown environments [41].

On the other hand, the task can also require considerable tracking accuracy along the motion vector, and at the same time flexibility along the other directions so as to adapt to unpredicted obstacles or external disturbances. In these cases, the impedance parameters need to be tuned in such a way they are stiff in the direction of the motion, and compliant along the other axes. This is achieved by tuning the major axis of the Cartesian stiffness and damping ellipsoids in the direction of interaction:

$$\boldsymbol{K}_c^{des} = \boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{U}^T, \ \boldsymbol{D}_c^{des} = \boldsymbol{U}\boldsymbol{\Sigma}_d\boldsymbol{U}^T, \tag{8}$$

where the diagonal matrix $\boldsymbol{\Sigma}_k = diag(k_{high}, k_{low}, k_{low})$ and $\boldsymbol{\Sigma}_d = diag(d_{high}, d_{low}, d_{low})$ elements are respectively the desired stiffness and damping coefficients along the direction of the vectors composing the $\boldsymbol{U}$ basis. The $\boldsymbol{U}$ columns are the basis vectors, the first one represents the desired motion vector, and the other two are shaped to form an orthonormal basis [41]. The sketch in Fig. 6 clarifies graphically this concept: being stiff along the motion vector and compliant in the other directions makes the robot adapt if an unexpected obstacle is found.

To ensure smooth operations and to avoid sudden jumps when switching between adapation mode and free space, the impedance matrices change smoothly according to the following law:

$$\boldsymbol{K}_c = f * \boldsymbol{K}_c^{des} + (1.0 - f) * \boldsymbol{K}_c$$

$$\boldsymbol{D}_c = f * \boldsymbol{D}_c^{des} + (1.0 - f) * \boldsymbol{D}_c \tag{9}$$

where f is a filter, set with a low value and depending on the control loop frequency, and $\boldsymbol{K}_c^{des}$ and $\boldsymbol{D}_c^{des}$ are respectively the Cartesian desired stiffness and damping matrices computed in (8).

3.4. Trajectory planner

The presented trajectory planner offers two different types of trajectories. The first one is a classical *point-to-point* motion, computed by means of a fifth-order polynomial, that given a starting and a target pose, computes the intermediate waypoints. The second one is given by a *contact-detection* motion, that when receives as input a unit vector and a threshold force, starts moving the robot end-effector from the current pose towards the unit vector direction until the forces along that direction go beyond a given threshold. This unit, which continuously reads the robot status, also regulates the switching between the Finite State Machine states by triggering motion-ending acknowledgments.

3.5. Pick and place Finite State Machine

The design of a Finite State Machine (FSM) is needed to regulate the data exchanged among the presented modules. Every state communicates with the *Trajectory planner*, asking for a desired motion (*point-to-point* or *contact-detection*) and receiving a *motion ended* acknowledgment, that serves as input to move on to the next state.

The FSM primitive motions can be summarized in three main states: pick an item from the conveyor, move it to a starting pose in the box, and place it towards the border/other items. "Pick from conveyor" sends a data request to the perception module "Conveyor localization" subunit and waits until the conveyor pose is sent back. Notice that, since a successful box detection is given only when static objects are perceived, if the human agent is simultaneously placing some items, the robot pauses its motion until he/she moves away. This enhances the framework's safety for the workers.

Once the conveyor pose is retrieved, the FSM sends to the *Trajectory planner* first a request for a *point-to-point* motion that leads the robot end-effector above the conveyor, and then a *contact-detection* motion that moves the robot down until a contact with the conveyor is detected. The robotic gripper can now grasp the item.

Note that, while carrying out the *Items reorganizing strategy*, the "Pick from conveyor" state is substituted by a similar motion that instead of picking the object from the conveyor, picks it directly from the box (exploiting the item pose detected by the perception unit), as described at the end of Sec. 3.2.

Next, in the "Move to start pose" state, the robot needs to reach the pose identified in Alg. 1 as ***start***. To do so, a data request is sent to the *Box and items detection* unit, that returns the current box **OG**, containing the data of the probability occupancy of every cell grid, its height, width and resolution, and the origin of the box reference frame Σ_B. This information, along with the item diameter, are then sent to the *Items placing strategy* module that returns the ***start*** pose and the unit vector $\hat{\boldsymbol{u}}$, that are translated in the robot frame through the transformation:

$$\boldsymbol{T}_I^R = \boldsymbol{T}_C^R \boldsymbol{T}_B^C \boldsymbol{T}_I^B \tag{10}$$

where $\boldsymbol{T}_I^R$ represent the transformation from the item reference frame $\boldsymbol{\Sigma}_I$ to the robot reference frame $\boldsymbol{\Sigma}_R$ (see Fig. 1).

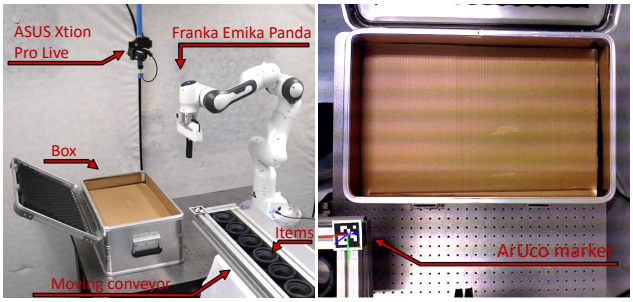

Figure 7: The experimental setup (a) includes a Franka Emika Panda robotic arm, an ASUS Xtion Pro Live camera, a moving conveyor with motor shells, and a box placed on the robot workbench. The view from the camera (b) shows the box and the detected *ArUco* marker sticked to the moving conveyor.

The robot first moves above the ***start*** pose with a *point-to-point* motion, and then down until a contact with the box is detected (with a *contact-detection* motion), as in the previous state. Finally, in the "Place" state, the robot moves towards the direction given by $\hat{\boldsymbol{u}}$ with a *contact-detection* motion until a contact with the border or other items already present in the box is found. This is done activating the *impedance adaptation* strategy along the direction of the motion, so as to ensure a more efficient item placing. This processes is repeated until there are items on the conveyor and/or there is a box with empty occupancy subgrids available.

4. Experimental setup

The presented software architecture, depicted in Fig. 4, relies upon the robotics middleware Robot Operating System (ROS) using `C++` as client library. The modules introduced in Sec. 3 are implemented as ROS nodes, and the data among them are exchanged through ROS services and messages.

To perform experiments, we used a Franka Emika Panda robotic arm equipped with its original two-finger gripper, that was modified with longer fingers to allow bigger objects grasping. The communication with the robot controller was established through the Franka Control Interface (FCI), that provides the current robot status and enables its direct con-

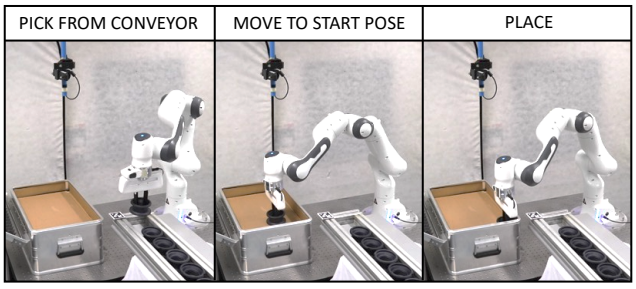

Figure 8: Experimental snapshots of the three Finite State Machine primitives.

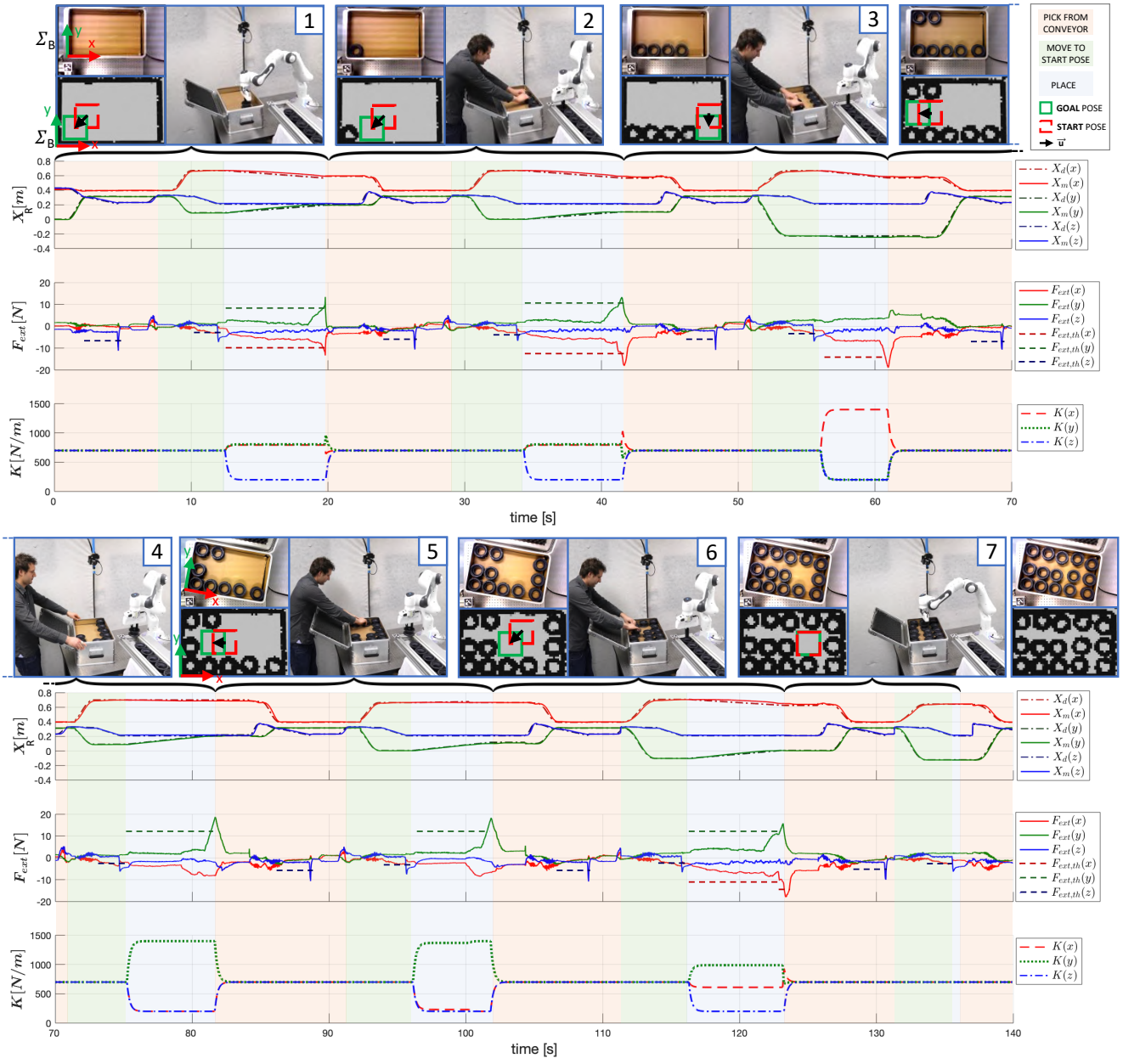

Figure 9: The robot and the human agent fill in the box in collaboration. The plots show the desired and measured position in the robot reference frame $\mathbf{\Sigma}_R$, the sensed external forces, and the stiffness profiles, along the Cartesian axes. The experimental snapshot, the camera view, and the occupancy grid are shown above the plots.

trol with an external workstation PC connected via Ethernet in real-time at a communication rate of 1 kHz. The perception data were streamed at a frequency of 30 Hz through an ASUS Xtion Pro Live RGB-D sensor, that was mounted above the robot workspace facing downward, so as to have a top view of the workspace, and calibrated with respect to the robot base frame. As items, we used industrial actuator shells.

5. Experiments

To validate the proposed method, we carried out experiments replicating a pick and place industrial scenario, that involves human-robot collaboration. To describe the different task phases, we follow the FSM flow through its three main states. The experimental snapshots of these states are depicted in Fig. 8: "Pick from conveyor", "Move to start pose", and "Place". In the last paragraph of this section, we show the results obtained with the *Items reorganizing strategy*, where the "Pick from conveyor" state is substituted with a similar motion primitive that picks the items directly from the box, in order to reconfigure them. In all the proposed experiments, the ***OG** resolution* was set to 1 cm, and f in (9) to 0.005.

Fig. 9 shows the data associated to an experiment that was carried out to fill in an entire box in collaboration with a

human subject. The three subplots represent the desired and measured position of the robot end-effector, the sensed external forces, and the stiffness profiles, along the three Cartesian axes. For every iteration of the three FSM states, above each relative plot, we represent the state of the **OG** taken during "Move to start pose" phase, and the snapshot of the robot and the human placing the items related to the "Place" state. The first one is composed by the raw image recorded by the RGB-D camera placed above the robot, and the **OG** retrieved by the *Box and items detection* module with that image. On the grid we also represent the data computed by the *Items placing strategy* following the nomenclature of Alg. 1: the green square represents the ***goal*** pose, the red dashed square identifies the ***start*** pose, and the black arrow constitutes the unit vector $\hat{\boldsymbol{u}}$.

In this scenario, to completely fill in the box, the FSM states were iterated 7 times, as numbered in the snapshots associated to the plots. In every iteration we can distinguish the three FSM states: "Pick from conveyor" is highlighted in light red, "Move to start pose" in light green, and "Place" in light blue. The first phase is similar for all the iterations, the robot reaches the conveyor pose and moves down on the *z-axis* until a contact with the conveyor is detected, i.e. when $||\boldsymbol{F}_{ext}(z)|| \geq \boldsymbol{F}_{ext,th}(z)$. The threshold $\boldsymbol{F}_{ext,th}(z)$ was set to 3 N, each time added to the different force bias sensed at the beginning of the relative motion. On the other hand, the second and the third phase change at every iteration, therefore hereafter we provide a complete description of all of them:

Iteration 1: the computed ***goal*** pose lies on the bottom-left corner, since the box is empty and the algorithm starts taking into account the ***OG*** cells with row and column equals to 0. There are 3 feasible *candidateStartGrids*, but the one from the top and the one from the right are assigned with a high weight since they are close to the grid border. On the contrary, the selected ***start*** pose has $weight = 0$ since the cells on its outer border are all empty. The item, after being moved to the ***start*** pose, is pushed against the borders until a contact on x and y is detected ($time = 20s$). The threshold $\boldsymbol{F}_{ext,th}(x, y)$ was set to 10 N, projected along the motion vector direction. Therefore in this case $\boldsymbol{F}_{ext,th}(x) \approx \boldsymbol{F}_{ext,th}(y) \approx 7N$ (also here added to the different force bias sensed at the beginning of the relative motion). In the third subplot, the stiffness profile adaptation is depicted, the robot is stiffer along the motion vector and more compliant in the other directions. In (8), k_{high} was set to $1400N/m$ and k_{low} to $200N/m$.

Iteration 2: similar to the previous iteration. In addition, a subject starts to add items in the box as well, as can be seen from the picture. The 2 items added by the human are also visible in the next iteration, where the detected ***OG*** includes 4 items.

Iteration 3: the computed ***goal*** pose lies on the bottom-right corner, and the only feasible *candidateStartGrid* is represented by the depicted ***start*** pose. The "Place" motion is only along the robot x axis, as can be noticed also in the impedance parameters regulation. Being compliant on the other axes ensures the success of the motion, since the item can be inserted in the empty spot regardless of any potential obstruction, given either by the other item or by the right border. In parallel, the subject places other 2 items in the top-left part of the grid.

Iteration 4: similarly as in iteration 3, only one feasible *candidateStartGrid* is found by the *Items placing strategy*. This time, the motion takes place only along the robot y axis. At the end of this phase, the subject moves the position of the box on the workbench. This has been done to show the flexibility of the framework, robust to changing conditions. In fact, the box pose is computed at the beginning of every iteration, and it is part of the information associated to the ***OG***.

Iteration 5: the *Items placing strategy* outcome is similar to the one computed in the previous iteration. In addition, here we can see that the stiffness parameters are tuned also along the robot x axis, since, despite the motion on the grid looks identical, the box has been rotated. During the "Place" state, the human puts other 2 items on the grid top-right.

Iteration 6: the computed unit vector $\hat{\boldsymbol{u}}$ is similar to the one of the first 2 iteration. However, due to the previously mentioned box rotation, the direction of the motion in the world frame is quite different. At the end of the "Place" state ($time \approx 123s$), we can see that the contact on y is detected slightly before the one along the robot x axis, so $\boldsymbol{K}(x)$ increases and $\boldsymbol{K}(y)$ decreases. Also $\boldsymbol{F}_{ext,th}(x)$ raises since, once the contact along y is detected, the motion on that direction is stopped and continues only along x. At the end of the robot motion, the subject places other 2 items in the box.

Iteration 7: only one empty spot is left on the grid. In the *Items placing strategy*, since no *candidateStartGrid* is found, the ***start*** pose is set equal to the ***goal*** pose. The third state, "Place", involves only the item ungrasping, since, after the contact detected on the z axis during the previous state, there is no need to move it on the xy plane.

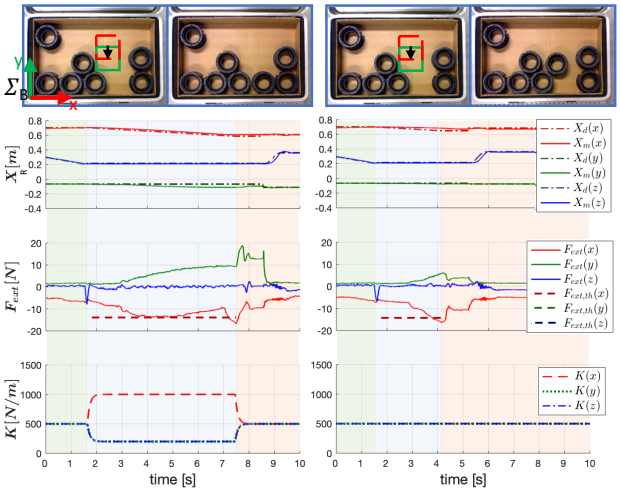

Figure 10: Starting from the same ***OG*** initial condition, the same experiment repeated twice: on the left with the impedance parameter adaptation, on the right without. Being stiffer along the motion vector, and more compliant in the other directions, improves the complete filling of empty spaces.

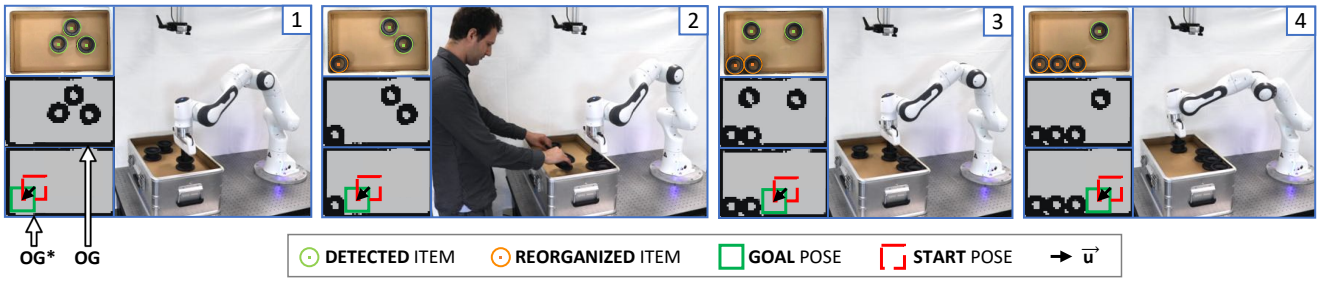

Figure 11: The *Items reorganizing strategy* allows the robot to compact items already lying on the box layer. For every iteration, the figure shows the raw camera image with the detected items, and with the subset of already reorganized ones (top-left), the original occupancy grid $\boldsymbol{OG}$ (middle-left), the modified occupancy grid $\boldsymbol{OG}^*$ computed through Alg. 2 (bottom-left), and snapshot of the human-robot team carrying out the relative task phase (right).

Comparison to the stiff controller: to better show the advantages of the online impedance regulation along the motion vector, we performed an additional experiment starting from the same condition, with and without the impedance parameters adaptation. Fig. 10 shows on the left the experiment carried out with the impedance regulation enabled, as can be seen from the third subplot. The placing motion does not stop where the $\boldsymbol{goal}$ pose was estimated, but it continues along the robot x axis until the border of the box is found. Being compliant on the y axis makes the robot insert the piece between the two items in the box bottom-right. In the first subplot, we can notice that the desired motion on y does not change while the measured one moves away, thanks to the above-mentioned compliance. The two camera snapshots, taken before and after the plotted data, are shown above the plots.

On the other hand, when the online tuning of the impedance profiles is not active, the situation explained above does not hold anymore. When the estimated $\boldsymbol{goal}$ pose is reached, the sensed external forces on the robot x axis go beyond the threshold $\boldsymbol{F}_{ext,th}(x)$ $(t = 4s)$, and the item is released from the gripper.

Items reorganizing strategy: this experiment aims to show the robustness of the framework in reorganizing items that are already lying on the box layer. Fig. 11 illustrates four iterations of this task. *1)* Initially, three items are detected by the *Items detection* vision unit, and since none of them belongs to the *itemsReorganized* list (see Alg. 2), all the detected items are removed from $\boldsymbol{OG}^*$. The item is therefore placed in the box bottom left corner. *2)* Still, three items are detected, but one of them is classified as already *reorganized*, and thus is added to $\boldsymbol{OG}^*$. While the item is placed on the right of the former one, an operator adds a new item to the box and moves an existing one to a different pose, so as to show the framework's robustness to changes. *3)* Among the four detected items, two are classified as *reorganized* (and added to $\boldsymbol{OG}^*$), and another item is stacked to the right of the previous ones. *4)* Only one detected item is still recognized as not *reorganized*, so the robot picks it and places it close to the box bottom right edge. The relative plots showing the robot end-effector pose, sensed external forces, and stiffness parameters are omitted since they are comparable to the ones showed in Fig. 9.

6. Conclusion and Discussion

In this work, we proposed a novel manipulation framework for a box-filling task. The flexibility is at the core of the method: the items can be picked from a moving conveyor detected by the perception module, and placed in a box following the presented sorting strategy, inspired by the human motor behavior. Additionally, it includes an adaptive Cartesian impedance controller to regulate the impedance profiles along the motion vector to guarantee the best outcome in the items placing. We experimentally validated the presented framework in placing a batch of motor shells, that were brought by a moving conveyor, inside a box located on the robot workbench.

Although the presented method can already handle heterogeneous objects thanks to the agile strategy provided by the occupancy grid, multiple object shapes will be investigated in our future works. Similarly, multiple layers have not been take into account in this work, and they will be object of future developments. Nevertheless, it is worth to notice that once a layer has been entirely filled in, the robot stays in hold because no empty space is detected in the occupancy grid; as soon as another layer is placed in the box (by a human or another robot), the occupancy grid is again free and the robot automatically resumes the procedure. It is also worth to notice that the *Items placing strategy* handles non-convex items by considering as shape the smallest square that can contain the original concave one. In such cases, thanks to the *Adaptive Cartesian impedance controller* that makes the item slide until a certain force threshold is sensed, an item can be placed even in the concave side of another item that has already been sorted in the box, thus leading to a more compact packing. However, concave shapes will be objective of further investigations, so as to guarantee a more efficient packing, for instance by rotating the items before sorting them in the box.

In conclusion, the proposed human-robot collaboration framework can lead to several improvements in the box-filling task. Even if the performance of the task execution in terms of cycle-time can be less that human-only settings (i.e., two humans performing the task), the introduced strategy presents several advantages compared to the human-only workforce. In fact, the robotic system, which is also capable of working

autonomously, can operate continuously during the working off shifts (e.g., at nights and over the weekends). Furthermore, the advantage of relieving stress from human workers cannot be undervalued. In fact, lifting and moving several heavy objects for repeated periods can lead to severe work-related musculoskeletal disorders and absenteeism of the operators [2, 42, 43], which eventually can contribute to a significant reduction of operational efficiency.

References

[1] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kosuge, and Oussama Khatib. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, 42 (5):957–975, 2018.

[2] Wansoo Kim, Marta Lorenzini, Pietro Balatti, PD Nguyen, Ugo Pattacini, Vadim Tikhanoff, Luka Peternel, Claudio Fantacci, Lorenzo Natale, Giorgio Metta, et al. A reconfigurable and adaptive human-robot collaboration framework for improving worker ergonomics and productivity. *IEEE Robotics and Automation Magazine*, 2019.

[3] Gaël Humbert, Minh Tu Pham, Xavier Brun, Mady Guillemot, and Didier Noterman. Comparative analysis of pick & place strategies for a multi-robot application. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8. IEEE, 2015.

[4] Kensuke Harada, Tokuo Tsuji, Kazuyuki Nagata, Natsuki Yamanobe, and Hiromu Onda. Validating an object placement planner for robotic pick-and-place tasks. *Robotics and Autonomous Systems*, 62(10): 1463–1477, 2014.

[5] Patrik Fager, Martina Calzavara, and Fabio Sgarbossa. Kit preparation with cobot-supported sorting in mixed model assembly. *IFAC-PapersOnLine*, 52(13):1878–1883, 2019.

[6] Qihan Wu, Meng Li, Xiaozhi Qi, Ying Hu, Bing Li, and Jianwei Zhang. Coordinated control of a dual-arm robot for surgical instrument sorting tasks. *Robotics and Autonomous Systems*, 112:1–12, 2019.

[7] Carola Zwicker and Gunther Reinhart. Human-robot-collaboration system for a universal packaging cell for heavy electronic consumer goods. In *Enabling Manufacturing Competitiveness and Economic Sustainability*, pages 195–199. Springer, 2014.

[8] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.

[9] Mustafa Suphi Erden and Bobby Marić. Assisting manual welding with robot. *Robotics and Computer-Integrated Manufacturing*, 27(4): 818–828, 2011.

[10] Bitao Yao, Zude Zhou, Lihui Wang, Wenjun Xu, Quan Liu, and Aiming Liu. Sensorless and adaptive admittance control of industrial robot in physical human- robot interaction. *Robotics and Computer-Integrated Manufacturing*, 51:158–168, 2018.

[11] Luz Martínez, Javier Ruiz-del Solar, Li Sun, J Paul Siebert, and Gerardo Aragon-Camarasa. Continuous perception for deformable objects understanding. *Robotics and Autonomous Systems*, 118:220–230, 2019.

[12] Ady-Daniel Mezei, Levente Tamás, and Lucian Buşoniu. Sorting objects from a conveyor belt using active perception with a pomdp model. In *2019 18th European Control Conference (ECC)*, pages 2466–2471. IEEE, 2019.

[13] Ching-Yen Weng, Wanqi Yin, Zhong Jin Lim, and I-Ming Chen. A framework for robotic bin packing with a dual-arm configuration. In *IFToMM World Congress on Mechanism and Machine Science*, pages 2799–2808. Springer, 2019.

[14] J Krüger and D Surdilovic. Robust control of force-coupled human–robot-interaction in assembly processes. *CIRP annals*, 57(1):41–44, 2008.

[15] Yang Chen, Shaofei Guo, Cunfeng Li, Hui Yang, and Lina Hao. Size recognition and adaptive grasping using an integration of actuating and sensing soft pneumatic gripper. *Robotics and Autonomous Systems*, 104:14–24, 2018.

[16] Kurt Landau, Holger Rademacher, Herwig Meschke, Gabriele Winter, Karlheinz Schaub, Marc Grasmueck, Ingo Moelbert, Michael Sommer, and Jens Schulze. Musculoskeletal disorders in assembly jobs in the automotive industry with special reference to age management aspects. *International Journal of Industrial Ergonomics*, 38 (7-8):561–576, 2008.

[17] Luca Gualtieri, Erwin Rauch, and Renato Vidoni. Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review. *Robotics and Computer-Integrated Manufacturing*, 67:101998, 2020.

[18] Vincenzo Lippiello, Bruno Siciliano, and Luigi Villani. Interaction control of robot manipulators using force and vision. *International Journal of Optomechatronics*, 2(3):257–274, 2008.

[19] Andrea Cherubini, Robin Passama, Arnaud Meline, André Crosnier, and Philippe Fraisse. Multimodal control for human-robot cooperation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2202–2207. IEEE, 2013.

[20] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European journal of operational research*, 183(3):1109–1130, 2007.

[21] CS Chen, Shen-Ming Lee, and QS Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76, 1995.

[22] Silvano Martello, David Pisinger, Daniele Vigo, Edgar Den Boef, and Jan Korst. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software (TOMS)*, 33(1):7–es, 2007.

[23] Andrea Lodi, Silvano Martello, and Daniele Vigo. Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420, 2002.

[24] José Fernando Gonçalves and Mauricio GC Resende. A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500–510, 2013.

[25] Fan Wang and Kris Hauser. Stable bin packing of non-convex 3d objects with a robot manipulator. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8698–8704. IEEE, 2019.

[26] Fan Wang and Kris Hauser. Robot packing with known items and nondeterministic arrival order. *IEEE Transactions on Automation Science and Engineering*, 2020.

[27] Fabio Sgarbossa, Anita Romsdal, Finn H Johannson, and Torbjørn Krogen. Robot picker solution in order picking systems: an ergozoning approach. *IFAC-PapersOnLine*, 53(2):10597–10602, 2020.

[28] Mohamed El Amine Boudella, Evren Sahin, and Yves Dallery. Kitting optimisation in just-in-time mixed-model assembly lines: assigning parts to pickers in a hybrid robot–operator kitting system. *International Journal of Production Research*, 56(16):5475–5494, 2018.

[29] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[30] Peter W Battaglia and Paul R Schrater. Humans trade off viewing time and movement duration to improve visuomotor accuracy in a fast reaching task. *Journal of Neuroscience*, 27(26):6984–6994, 2007.

[31] Paul M Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954.

[32] Roland Arsenault and Colin Ware. Eye-hand co-ordination with force feedback. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 408–414, 2000.

[33] Daniel Goldreich and Ingrid M Kanics. Tactile acuity is enhanced in blindness. *Journal of Neuroscience*, 23(8):3439–3445, 2003.

[34] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Rafael Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481–491, 2016.

[35] Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76:38–47, 2018.

[36] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[37] J. Illingworth and J. Kittler. The adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9 (5):690–698, 1987. doi: 10.1109/TPAMI.1987.4767964.

[38] Christopher Schindlbeck and Sami Haddadin. Unified passivity-based cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 440–447, 2015.

[39] Arash Ajoudani, Nikos G Tsagarakis, and Antonio Bicchi. Choosing poses for force and stiffness control. *IEEE Transactions on Robotics*, 33(6):1483–1490, 2017.

[40] Pietro Balatti, Dimitrios Kanoulas, Giuseppe F Rigano, Luca Muratore, Nikos G Tsagarakis, and Arash Ajoudani. A self-tuning impedance controller for autonomous robotic manipulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5885–5891, 2018.

[41] Pietro Balatti, Dimitrios Kanoulas, Nikos G Tsagarakis, and Arash Ajoudani. Towards robot interaction autonomy: Explore, identify, and interact. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9523–9529. IEEE, 2019.

[42] Daria Battini, Martina Calzavara, Alessandro Persona, and Fabio Sgarbossa. Additional effort estimation due to ergonomic conditions in order picking systems. *International Journal of Production Research*, 55(10):2764–2774, 2017.

[43] Martina Calzavara, Christoph H Glock, Eric H Grosse, and Fabio Sgarbossa. An integrated storage assignment method for manual order picking warehouses considering cost, workload and posture. *International Journal of Production Research*, 57(8):2392–2408, 2019.