# Grasp It like a Pro: Grasp of Unknown Objects with Robotic Hands based on Skilled Human Expertise

Chiara Gabellieri[1,3], Franco Angelini[1,2,3], Visar Arapi[1], Alessandro Palleschi[1,3], Manuel G. Catalano[2], Giorgio Grioli[2], Lucia Pallottino[1,3], Antonio Bicchi[1,2,3], Matteo Bianchi[1,3] and Manolo Garabini[1,3]

*Abstract*— **This work proposes a method to grasp unknown objects with robotic hands based on demonstrations by a skilled human operator. Not only are humans efficacious at grasping with their own hands but are also capable of grasping objects using robotic hands. Therefore, we consider how the grasping skills of a human trained in robotic hand use can be transferred to a robot equipped with the same hand. We propose that a skilled human user manually operates the robotic hand to grasp a number of elementary objects, consisting of different boxes. A Decision Tree Regressor is trained on the data acquired from the human operator to generate hand poses able to grasp a general box. This is extended to grasp objects of unknown and general shape leveraging upon the state-of-the-art Minimum Volume Bounding Box decomposition algorithm that approximates with a number of boxes the point cloud of the object.**

**We report on extensive tests of the proposed approach on a Panda manipulator equipped with a Pisa/IIT SoftHand, achieving a success rate of 86.7% over 105 grasps of 21 different objects.**

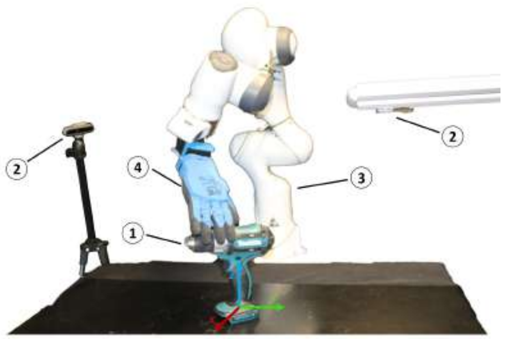


Fig. 1. The Panda manipulator executing a grasp on a power drill. The experimental setup is composed of: (1) the object to be grasped and the reference frame. The object is placed at a random position in the *xy* plane with a random orientation along the *z* axis; (2) two Intel RealSense Depth Cameras D415 employed to sense the object; (3) a Panda manipulator by Franka EMIKA; (4) the employed end-effector, the Pisa/IIT SoftHand.

## I. INTRODUCTION

Grasping previously unseen objects represents an open and very challenging problem for robots [1]. Traditional grasp theory focused mostly on searching contact points on the (usually known) object in order to satisfy some constraints and typically guarantee force closure [2]. In [3] these methodologies have been defined as *analytic* to distinguish them from the *empirical* (or *data-driven* [1]) ones, which have gained a greater attention in the latest years.

The efficacy of the data-driven approaches depends on the quality of the grasp data sets. A prominent example is the Cornell Grasping Dataset, containing numerous objects and ground-truth labeled grasps specifically designed for parallel grippers. This data set allows the training of grasp-detection deep neural networks (e.g., [4], [5], and [6]). The adoption of such an approach for a different hand should rely on the creation of an extensive sample set, which is a time consuming and not trivial task.

On the other side, studies on vacuum grippers [7] and parallel grippers [8], [9] use synthetic training data sets in order to reduce data collection time [8]. Such data sets rely on a model of the end-effector.

Unfortunately, a reliable and computationally efficient simulator is not available for all types of robotic hands, limiting the applicability of model-based approaches. This is the case, e.g., for new robotic hands designed to structurally embody the *intelligence* of the human hand, through compliance and/or under-actuation [10]. An effort to model such robotic hands is the simulator described in [11]. However, this approach still may suffer from not accurately modeled effects such as friction generated by the gloves that usually cover the hands, and the contacts between the objects and the hand, which can be many more than 2 in case of adaptive hands. Another issue is the computational cost of simulating a system with a large number of bodies (e.g., 20 in case of the Pisa/IIT SoftHand [12]) or continuously soft bodies [13].

A different approach has been followed in [14], where the authors collect data from videos of humans grasping objects with their hands; then, they propose a reactive grasp planner for soft hands exploiting feedback reflexes triggered by a sensorized glove.

In [15] and [16] the authors generate very slim databases that associate the local shape of the object to each grasp of the training set. Then, the robot grasps previously unseen objects that *locally* resemble one of the shapes in the

This paper was recommended for publication by Editor H. Liu upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the European Unions Horizon 2020 Research and Innovation Program under Grant Agreement No. 732737 (ILIAD) and No. 840446 (SoftHandler), and in part by the Italian Ministry of Education and Research in the framework of the CrossLab project (Departments of Excellence).

[1]Centro di Ricerca "Enrico Piaggio", Università di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy

[2]Soft Robotics for Human Cooperation and Rehabilitation, Fondazione Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy

[3]Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy

chiara.gabellieri1@gmail.com

Digital Object Identifier (DOI): see top of this page.

database. The success of these promising approaches still depends on the variability of the database.

The approximation of object point clouds with basic shapes has been exploited in order to make grasping of unknown objects more easily tractable. In [17] a method to grasp an object by representing it with one superquadric is presented. However, the approximation of the entire object shape with a single superquadric is often not sufficiently accurate [18]. In [19] the authors localize graspable parts of object point clouds that the robotic hand can completely wrap. These graspable regions could also be further approximated with bounding cylinders. The method is tested on a parallel gripper generating grasps based on geometric considerations. In [18] the authors state that, in order to grasp unknown objects, the MVBB (Minimum Volume Bounding Box) decomposition is an effective trade-off between good approximation and efficiency. MVBB algorithm has been tested in simulation as a tool for grasp planning in [20]: given the box decomposition of an object, a pose is heuristically computed based on user-defined geometric features, and random variations of the defined grasping pose are tested. In this work, we aim to perform grasps of unknown single objects using robotic hands. Our major contribution is a data-driven planner that generates suitable grasps by relying on a reduced database of grasps performed by a skilled operator using the robotic hand. With ground-truth grasps provided by humans, which have already shown themselves capable of grasping with robotic hands (see e.g. [21]), it is not necessary to rely solely on precise geometric considerations for generating grasps. Besides, a key aspect of our approach is a simple method to create a very slim grasp database. The idea is to exploit a skilled human performing experiments using the robotic hand to grasp only a set of basic shapes instead of general objects, dramatically reducing the number of trials. This approach is then generalized to grasp unknown objects by relying on state-of-the-art decomposition algorithms that allow approximating an object with such basic shapes.

Specifically, the human operator grasps cuboid boxes, and MVBB decomposition algorithm proposed in [18] is used to decompose the object point clouds into bounding boxes. Hence, when a new object is presented to the robot, its point cloud is collected through RGB-D cameras and decomposed into bounding boxes; one of the boxes is selected as the best one, based on defined indexes. Given a candidate box, a Decision Tree Regression (DTR) algorithm trained on the human data predicts how a professional user would grasp a generic cuboid with the robotic hand, thus providing a set of suitable grasps. The predicted grasps are successively ranked and checked for collision avoidance. Eventually, a robotic manipulator, equipped with the same robotic hand that the human operator used to generate the database, performs the first candidate grasp.

With this method a general unknown object can be automatically grasped after a reduced training phase based only on sample boxes. We chose box approximation based on the availability of efficient state-of-the-art MVBB algorithms, purposefully designed in [18] as an aid for robotic grasping. Potentially, a different type of approximating shape could be used for the method implementation. Various shapes together might be used at the expense of increased complexity of human grasps acquisition and point cloud decomposition.

Employing a manipulator equipped with Pisa/IIT SoftHand (Fig. 1), we extensively tested the proposed approach on a set of 21 objects previously unseen by the regressor. We performed five tests of grasp for each single object, placed in a tabletop configuration, for a total of 105 grasps, obtaining an overall percentage of grasp success of $86.7\%$.

## II. Grasp Planning Algorithm

This section presents the method used to generate a grasping pose. The entire procedure is outlined in Algorithm 1 and explained in the following. The main steps of the algorithm and their outcomes are shown in Fig. 2 and Fig. 3, respectively.

The algorithm starts from the acquisition of the object point cloud (see Fig.3(a)), referred to as $PointCloud$. Once the point cloud is obtained (white block labeled "Perception" in Fig. 2), the function *MVBB()* computes an approximation of the object given by $N$ cuboid boxes (see Fig. 3(b)), contained in the list $Boxes$. The user-defined parameter of the function, $t$, is related to the minimum volume of the bounding boxes, see [20] for more details. To obtain the bounding box decomposition from the object point cloud (yellow block labeled "MVBB Algorithm" in Fig. 2), we use the procedure described in Algorithm 1 and Algorithm 2 in [20] based on [18]. Each box contained in $Boxes$ is described through 9 parameters: 3 for the dimensions, and 6 for the pose. The variable $SortedBoxes$ contains the same data as $Boxes$, but ranked according to *BoxSorting()* function, detailed in Algorithm 2. The first item of $SortedBoxes$ is the candidate box to be grasped (see Fig. 3(c)).

At this stage, the function *DTRPrediction()* predicts 48[1] possible poses to grasp the candidate box $BoxCand$ (see Fig. 3(d)). The poses are stored in the variable $Grasps$. The function is based on a Decision Tree Regressor trained on the data registered from the skilled human operator (green block labeled "DTR" in Fig. 2), and it is better detailed in Sec. II-B. Eventually, the function *GraspSelection()* selects a candidate grasp pose, namely $GraspCand$, among the ones stored in $Grasps$ (see Fig. 3(e)). It is possible that no feasible grasp can be selected among the ones in $Grasps$, e.g. because all the grasps would result in the hand colliding with the environment or with the object. In this case, the procedure is repeated from the point in which *DTRPrediction()* is called, assigning to $BoxCand$ the second box contained in $SortedBoxes$, and so on until a feasible grasp is selected or until all the boxes are considered. The proposed grasp selection algorithm follows a greedy approach, which consists of selecting one single box and evaluating the quality of the grasps for that box. Such a sub-optimal approach has the

[1]For each face, two grasps are predicted along the short side and two along the other side. Due to the symmetry of the boxes, each grasp is then rotated by $180\,\mathrm{deg}$ around the axis normal to the box face to consider the other approaching directions.

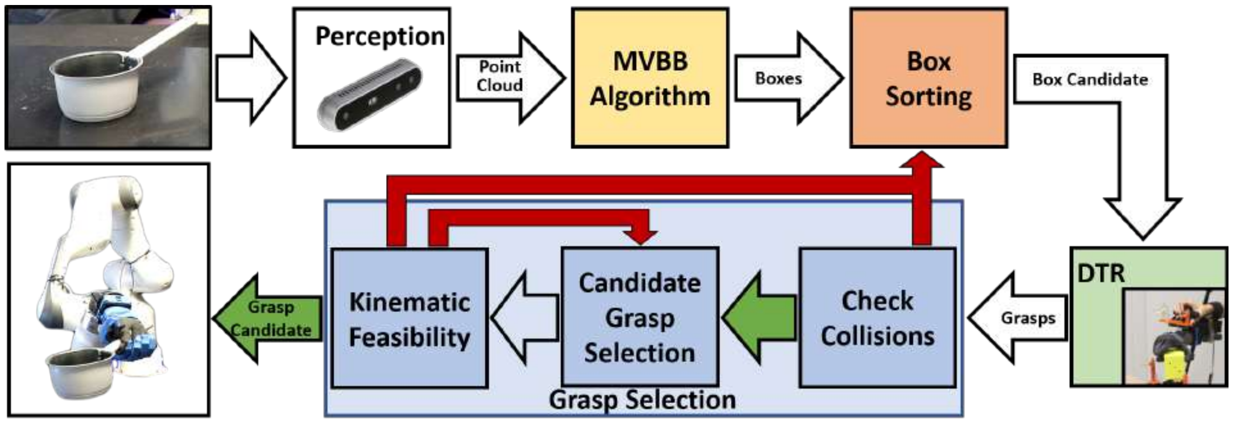
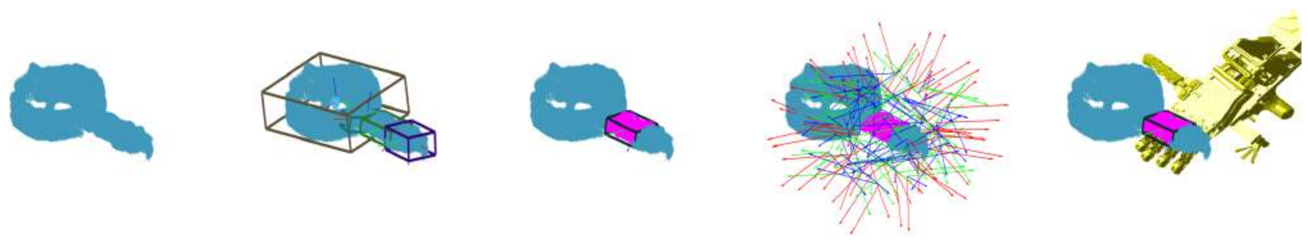


Fig. 2. The diagram summarizes the basic steps of the procedure. First, the point cloud of a new object is acquired. Then, it is processed by a bounding box algorithm (MVBB) with approximating cuboid boxes. A candidate box is selected and a DTR algorithm, trained on data of a skilled human operator grasping boxes using the robotic hand manually, predicts the grasp poses to grasp the selected box. A collision avoidance algorithm discards the unfeasible grasps, and among the feasible ones, a candidate is selected. If none of the graps is feasible for the candidate box, another box is selected. Inverse kinematics is used to check the kinematic feasibility of the pose based on the kinematic model of the robot used to perform the grasp. If the candidate grasp is not kinematically feasible, another grasp is selected. If none of the grasps is feasible, another box is selected. Eventually, the robot performs the grasp.



(a) Point Cloud. (b) Boxes (N=4). (c) Box candidate. (d) Grasps. (e) Grasp candidate.

Fig. 3. This sequence shows the outputs of the blocks in Fig. 2, labelled as the corresponding arrow in the diagram of in Fig. 2.

advantage of a low computational cost also with numerous boxes.

Eventually, $GraspCand$ is assigned as desired pose for the robotic manipulator by the function *SendToRobot()*. If no feasible grasp has been found for any box, then the procedure fails. This occurs in the unlikely eventuality that none of the boxes is graspable by none of the 48 grasps associated to it. However, a possible recovery policy is to restart from line 2 of Algorithm 1 with a different value of the parameter $t$. This leads to a different box decomposition, which may result in a non-null feasible grasps set.

## A. Candidate Box Selection

Algorithm 2 outlines the steps of the candidate box selection, namely the content of the orange block labeled "Box Sorting" in Fig. 2. The boxes contained in $Boxes$ are ranked by decreasing values of the performance index, $P$, and stored in $SortedBoxes$.

In this work, a box ranking policy that favors the outermost box has been adopted, as done in [20]. Additionally, noise can introduce scattered points in the point cloud leading to bounding boxes with a low density of points. If grasped, such boxes may result in a grasp that does not have a grip on many actual points of the real object. To address this issue, our policy also favors the boxes with a higher density

**Algorithm 1** Object grasping procedure

```
1:  Get $PointCloud$;
2:  $Boxes$ = MVBB($PointCloud, t$);
3:  $SortedBoxes$ = BoxSorting($PointCloud, Boxes$);
4:  $GraspCand = [\ ]$;
5:  $i = 1$;
6:  while ($GraspCand == [\ ]$ && $i \leq N$) do
7:      $BoxCand = SortedBoxes[i]$;
8:      $Grasps$ = DTRPrediction($BoxCand$)
9:      $GraspCand$ = GraspSelection($PointCloud$,$Grasps$,
    $BoxCand$);
10:     $i = i + 1$;
11: SendToRobot($GraspCand$);
```

of points. The $i-$th element of $Density$, indicated with $Density[i]$, contains the ratio between number of points of the point cloud $PointCloud$ contained inside the $i-$th box, $Boxes[i]$ and $Volume[i]$, namely the volume of $Boxes[i]$. $NormDens[i]$ is the $Density[i]$ normalized by the maximum value of the vector $Density$ (computed by *Max()* function). The volume of a box, given its dimensions, is returned by *GetBoxVolume()*. The number of points contained in a box, stored in $Points$, are computed by *PointsInside()*.

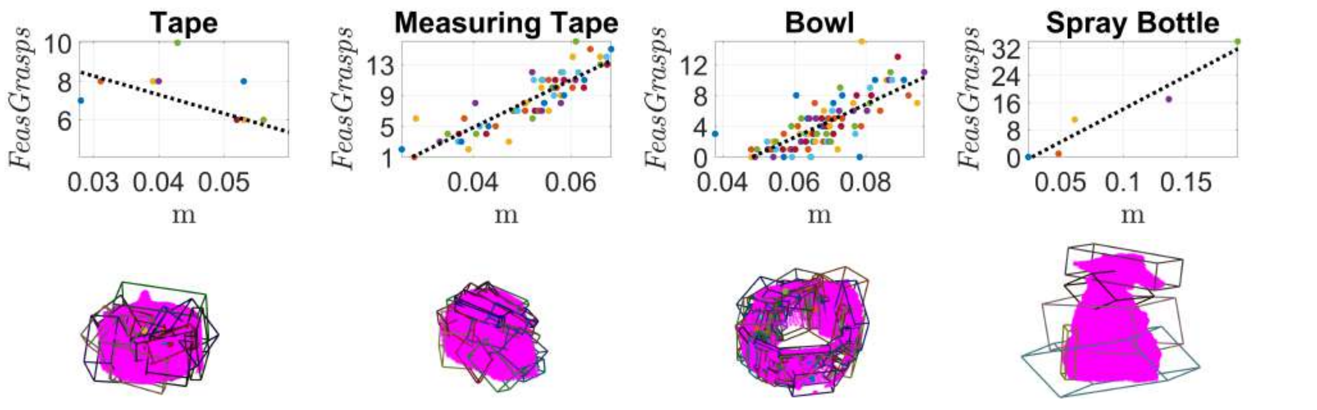


Fig. 4. Number of feasible grasps versus distance from the centroid of the point cloud (non-normalized, expressed in meters) for four objects from Tab. I. A linear fitting of the data is shown by a black dotted line. Below each plot there is the box decomposition of the corresponding object.

**Algorithm 2** Box Sorting
```
1: function BoxSorting($PointCloud$,$Boxes$)
2:     $Centroid$ ←CompCentroid($PointCloud$);
3:     for ($i = 1$, $i \leq N$, $i++$) do
4:         $Volume[i]$ = GetBoxVolume($Boxes[i]$);
5:         $Points[i]$ = PointsInside($Boxes[i]$, $PointCloud$);
6:         $Density[i] = Points[i]/Volume[i]$;
7:         $Distance[i]$ = CompDist($Centroid$, $Boxes[i]$);
8:     for ($i = 1$, $i \leq N$, $i++$) do
9:         $NormDens[i] = Density[i]/Max(Density)$;
10:        $NormDist[i] = Distance[i]/Max(Distance)$;
11:        $P[i] = \frac{1}{2}(NormDens[i]^2 + NormDist[i]^2)$;
12:    $SortedBoxes$ = Sort($Boxes$, $P$);
13:    return $SortedBoxes$;
```

The function *CompDist()* computes the distance between the center of the $i-$th box and the centroid, $Centroid$, of the point cloud. The centroid is the arithmetic mean of all the positions of the points and is computed by *CompCentroid()*. $NormDist[i]$ contains the value of $Distance[i]$ normalized by the maximum value of the vector $Distance$.

## B. Exploitation of acquired human expertise

This section explains the content of the green block labeled "DTR" in Fig. 2, and of the function *DTRPrediction()* in Algorithm 1, which associates to $BoxCand$ the grasp poses of the hand. This is done by learning a model $\mathbf{y} = f(\mathbf{x})$ capable of predicting a human-like hand pose $\mathbf{y} = [y_1, ..., y_6]^T \in \mathbb{R}^6$, given the box dimensions $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$. To this end, we employ a Decision Tree Regressor (DTR). DTR methods are indeed particularly suited for reduced-dimension training sets, as it is our case. The representation of the Regression Tree model is a binary tree, where each node represents a single input variable $x_j$, with $j = \{1, .., 3\}$ and a split point on that variable. The leaf nodes of the tree contain an output variable $\mathbf{y}$ which is used to make a prediction. Creating a binary decision tree is actually a process of dividing up the input space. A greedy approach is used to divide the space called *recursive binary splitting*. Given an observation $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1, 2, \ldots, n$, the regression tree construction is explained by the following steps: i) select a splitting variable $j$ and a split point $s$; ii) define 2 regions $R_1$ and $R_2$: $R_1(j, s) = \{\mathbf{x}|x_j \leq s\}$, $R_2(j, s) = \{\mathbf{x}|x_j > s\}$; iii) for each $k \in \{1, \ldots, 6\}$ seek the splitting variable $j$ and the split point $s$ that solve

$$\min_{j,s}[\min_{c_{1,k}} \sum_{\mathbf{x}_i \in R_1(j,s)} (y_{i,k} - c_{1,k})^2 + \min_{c_{2,k}} \sum_{\mathbf{x}_i \in R_2(j,s)} (y_{i,k} - c_{2,k})^2] ,$$

where $c_{1,k}, c_{2,k} \in \mathbb{R}$ are two constant decision variables that describe the model response. Given $j$ and $s$, the solution of the minimization is

$$\hat{c}_{1,k} = \text{ave}(y_{i,k}|\mathbf{x}_i \in R_1(j, s)), \hat{c}_{2,k} = \text{ave}(y_{i,k}|\mathbf{x}_i \in R_2(j, s)),$$

where ave is the average of $y_{i,k}$ in region $R_1$ or $R_2$, and we define $\hat{c}_1 = [\hat{c}_{1,1}, \hat{c}_{1,2}, \ldots, \hat{c}_{1,6}]^T$ and $\hat{c}_2 = [\hat{c}_{2,1}, \hat{c}_{2,2}, \ldots, \hat{c}_{2,6}]^T$. iv) after finding the best split for each splitting variable, split the data into the 2 regions and repeat the splitting process recursively on each of the two regions. The maximum tree depth is empirically set equal to 8 as a trade-off between model complexity and minimization of overfitting risk. We used *SciKit-Learn* to train the algorithm, using as labeled dataset the one described in Sec. III. Hold out validation has been used to verify the generalization and robustness of pose prediction. We trained different model configurations to adjust the model parameters using the mean square error (MSE) between the predicted poses and the true labeled poses in the validation dataset.

## C. Candidate Grasp Selection

This section provides details about the content of the blue blocks collectively labeled as "Grasp Selection" in Fig. 2. This procedure, detailed in Algorithm 3, selects a candidate grasp once all grasps of $BoxCand$ are generated by the *DTRPrediction()* function. First, the function *CheckCollisions()* (right-hand side blue block in Fig. 2) discards all the grasps that would result in the hand colliding with the object and/or the environment. This is done approximating the hand by a virtual cuboid box. The function *PointsInside()* is called in order to compute how many points lie inside the hand box.

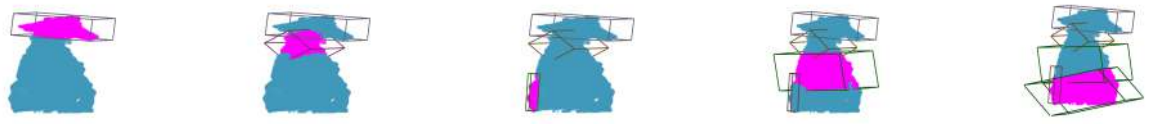
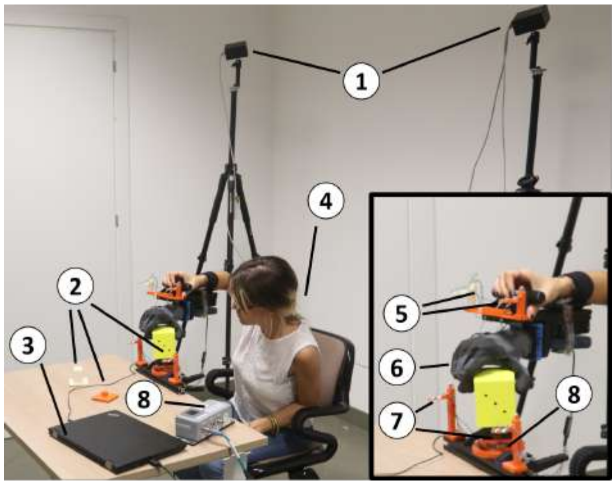

(a) First candidate. (b) Second candidate. (c) Third candidate. (d) Fourth candidate. (e) Last candidate.

Fig. 5. Box ranking obtained from Algorithm 2 applied to a point cloud decomposition for the Spray Bottle.

**Algorithm 3** Grasp Selection

```
1: function GraspSelection($PointCloud$, $Grasps$, $BoxCand$)
2:     $FeasGrasps$ = CheckCollisons($PointCloud$, $Grasps$, $EnvCloud$);
3:     if $FeasGrasps == [\,]$ then
4:         $GraspCand = [\,]$;
5:     else
6:         $GraspCand$ = CandGraspSelection($FeasGrasps$, $BoxCand$);
7:         while (KinFeasibility($GraspCand$)==false do
8:             $FeasGrasps$ = $FeasGrasps \setminus GraspCand$;
9:             $GraspCand$ = CandGraspSelection($FeasGrasps$, $BoxCand$);
10:    return $GraspCand$;
```

If the number of such points is greater than a threshold, then the corresponding grasp pose is discarded. Similarly, the collisions between the hand and the environment point cloud, $EnvCloud$, are checked. The remaining grasps are stored in $FeasGrasps$; if no grasp is collision-free, then $FeasGrasps$ is an empty variable. Eventually, among the remaining grasps, if any, the function *CandGraspSelection()* (central blue block in Fig. 2) selects the one with the thumb aligned with the longest side of $BoxCand$, according to what has been already explored in [20]. The boolean function *KinFeasibility()* (corresponding to the left-hand side blue block in Fig. 2), checks if $GraspCand$ is kinematically feasible for the robot. If $GraspCand$ is not kinematically feasible, then it is removed (through the operator $\setminus$) from the list $FeasGrasps$ and another grasp pose is selected as $GraspCand$. To support the choice of our box selection policy, we used the number of collision-free grasps contained in $FeasGrasps$ to show that the boxes more distant from the centroid of the point cloud are more accessible by the hand. For each object in Tab. I, we plotted the number of collision-free grasps for each box (random values of $t$ were used) as a function of the distance of the box from the centroid of the point cloud (containing also a portion of the table on which the object lies). Linear fittings of the data for all objects show an average slope of 15.45, which means that the number of feasible grasps associated with a box increases with the distance from the centroid. A negative slope has been obtained only for two objects: the Tape ($-6.57$), probably due to its symmetry and flatness, and the Courgette (-0.04), lying on the table. The highest values of slope have been obtained for the tallest objects: the Drill (28.35) and the Spray Bottle (37.05). See Fig. 4 for some examples. Figure 5 shows the results of our box ranking on one decomposition of the point cloud of the Spray Bottle. The first ranked box according to our policy is the farthest from the table (Fig. 5(a)). This solution is a reasonably desirable one since it likely leads to hand poses distant from the obstacle represented by the table. In the case of objects with handles or protrusions, these elements are also likely selected (see, e.g., Fig. 3(c)), which is another favorable choice in many cases. Note also that the third candidate box (Fig. 5(c)) is closer to the centroid of the point cloud than the fourth box (Fig. 5(d)). However, it has a better rank because it provides a better local approximation of the object point



(a)

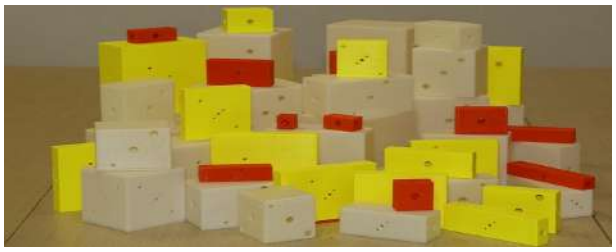

(b)

Fig. 6. Experimental setup during the first phase of the study. In 6(a), the acquisition of the data from an expert human user (4). The PhaseSpace cameras (1) record the position of the LED markers, eight world-fixed markers (7) and eight hand-fixed ones (5). A set of boxes (e.g. 2) are grasped with a Pisa/IIt SoftHand (6). The interaction forces during the grasp are measured with a torque-force sensor (8). All the data are recorded on a pc (3). In 6(b), the complete set of sample boxes.

cloud (higher density of points).

## III. Acquisition of human expertise

This section describes the first set of experiments, aimed to collect the data of the pose of the robotic hand used by a skilled human operator for grasping a set of sample boxes. The following setup has been employed:

- a Pisa/IIT SoftHand [12] provided with a handle and a battery to be used manually by the human operator;
- a commercial Phase Space Motion Capture System for 3D motion tracking with active LED markers, the Phase Space[2], to record kinematic data: ten stereo cameras working at 480 Hz tracked the 3D positions of 8 markers put on the handle of the Pisa/IIT SoftHand and of 8 world-fixed markers. The LED frequency is in the visible red;
- a fixed station made of two supports on which the 8 world-fixed LED markers are put, as shown in Fig. 6, equipped with a force-torque sensor ATI mini45, placed as in Fig. 6;
- a set of 56 cuboid sample boxes to be grasped (Fig. 6(b)), whose dimensions vary within the discrete interval $\{15, 30, 45, 60, 75, 90\}$mm. The smallest box is a cube of edge 15 mm, and the largest one is a cube of edge 90 mm. These bounds are dictated by the Pisa/IIT SoftHand design and are provided by the supplier[3].

Fig.6 shows the above-described experimental setup. The real experiments have been conducted in dark conditions. Each box has been rigidly fixed to the sensorized platform before being grasped. This allows measuring the interaction forces between the environment and the box itself (left for future use). The 8 world-fixed markers placed on their supports are used to robustly define a fixed reference frame. The 8 hand-fixed markers are placed on two supports on the handle, as shown in Fig. 6(5). They are used to define a reference frame placed on the palm of the Pisa/IIT SoftHand, which is fixed relative to the handle. A custom application developed in C++ enabled the synchronization between Phase Space data and force/torque sensor, and the Phase Space OWL library was used to get the optical tracking data.

Two different grasps, each repeated three times so to have multiple acquisitions, were performed on the different faces of each box, both along the short side and the other side, for a total of 648 trials.

## IV. Experimental validation

### A. Experimental Setup

The experimental setup is depicted in Fig. 1. The object to be grasped (1) is placed by an operator on the table. Both the object position along the *xy* plane and its orientation along the *z*-axis are randomly chosen by the operator. Two Intel RealSense Depth Cameras D415[4] (2) capture the point cloud of the object. Then the outcome of the proposed algorithm is used to grasp the object with an impedance controlled Panda arm by Franka EMIKA[5] (3). The manipulator is equipped with the same end-effector employed to acquire the human expertise (Sec. III), i.e., the Pisa/IIT SoftHand (4) and it is controlled using ROS.

[2]http://phasespace.com/

[3]https://qbrobotics.com/wp-content/uploads/2016/03/QBsofthand_datasheet_general.pdf

[4]https://www.intelrealsense.com/depth-camera-d415/

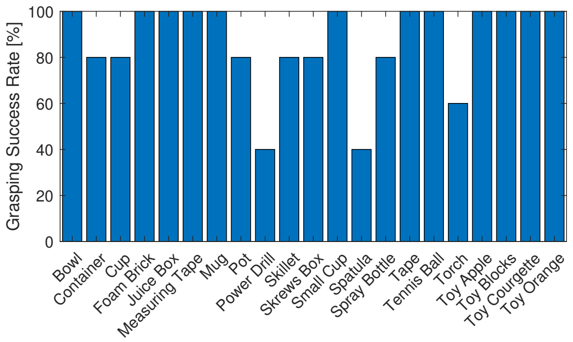


Fig. 7. Grasping success rate for each of the 21 objects, the total average is 86.7%. Each item has been tested 5 times.

To validate the method, we tested 21 objects. Tab. I shows the objects and reports their geometrical and physical properties. The objects were chosen to span a wide variety of size, weight, texture, and stiffness. Most of the objects present similar characteristics to the ones of the benchmark set proposed in [22], and none of them was used to train the DTR. Each object is tested 5 times with different random positions and orientations. This means that a total of 105 grasps have been tested. After closing the fingers, the end-effector of the robot is lifted of 150mm in 5.5s. If the object does not fall during this interval, the grasp is considered successful. The parameter of the bounding box algorithm [20] is fixed and equal to $t = 5 \cdot 10^{-5}$. This value is a trade-off between performance and computational time.

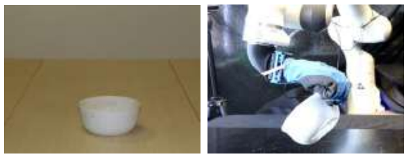
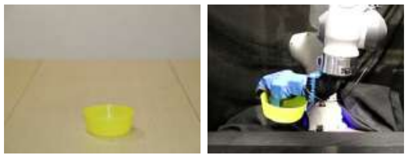
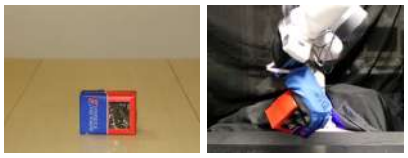
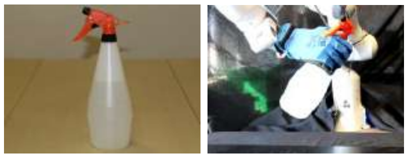
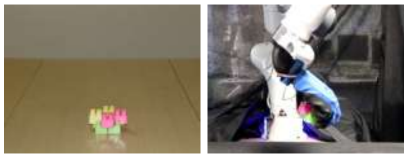
### B. Discussion

The results are reported in Fig. 7. The average grasping success rate, among all the 105 trials is equal to 86.7%. First of all, it is worth mentioning that the efficacy of the proposed method relies on the accuracy of the captured point cloud. If the object is only partially captured, or if the point cloud is noisy, then also the approximation of the object in minimum volume bounding boxes will be poor, resulting in a failure. The adaptability of Pisa/IIT SoftHand to the shape of the grasped object may play a role in determining the success of a grasp in the presence of a rough approximation of the object. Furthermore, at the moment we are considering a fixed parameter for the MVBB algorithm. Its value has been chosen empirically, after a preliminary visual inspection of the results of the box decomposition applied to the test objects. Automatic tuning of the parameter depending on the point cloud may be beneficial to the method, although its definition is not straightforward. Indeed, different objects with different dimensions and shape complexity can

[5]https://frankaemika.github.io/docs/

TABLE I

SET OF TESTED OBJECTS. EACH CELL REPORTS A PICTURE OF THE OBJECT, ITS SIZE, WEIGHT AND A SUCCESSFUL GRASP.

| | | |
|---|---|---|
| (a) Bowl, ∅140×65mm, 55g.  | (b) Container, ∅115×48mm, 35g.  | (c) Cup, ∅74×84mm, 35g. |
| (d) Foam Brick, 92×44×44mm, 6g. | (e) Juice Box, 60×38×80mm, 221g. | (f) Measuring Tape, 75×35×65mm, 189g. |
| (g) Mug, 135×95×100mm, 345g. | (h) Pot, 280×133×74mm, 374g. | (i) Power Drill, 200×95×195mm, 1190g. |
| (j) Skillet, 380×230×40mm, 316g. | (k) Screws Box, 125×55×90mm, 444g.  | (l) Small Cup, 80×50×60mm, 107g. |
| (m) Spatula, 310×80×60mm, 45g. | (n) Spray Bottle, 120×95×290mm, 927g.  | (o) Tape, ∅89×53mm, 62g. |
| (p) Tennis Ball, ∅71mm, 46g. | (q) Torch, 185×∅78mm, 466g. | (r) Toy Apple, ∅73×64mm, 140g. |
| (s) Toy Blocks, 87×87×47mm, 34g.  | (t) Toy Courgette, 145× ∅49mm, 15g. | (u) Toy Orange, ∅73mm, 18g. |

be better approximated with a larger number of boxes. A better approximation of the point cloud resulting from an automatic parameter tuning may be advantageous also for the application of the method to rigid hands. If different hands were used, it might be necessary to modify the human pose acquisition procedure, e.g., registering also the position of the fingers. Very effective data-driven methods for grasping with rigid hands have been proposed in the literature. However, they usually require huge data sets of object models and grasps [23], [8], or extensive experimental training [24], limiting their generality. In this work, we propose a method whose generality may be aided by its independence from object models resulting in a reduced data set.

Another aspect worth noting is that the proposed method evaluates only the feasibility of the final candidate pose and not of the approaching trajectory. A collision avoidance algorithm may be integrated to improve effectiveness.

Furthermore, the objects more challenging to grasp resulted to be the Spatula and the Power Drill. The difficulty of the former object is linked to its geometrical properties. Indeed, it is a flat object lying on the table, thus it may occur that the hand fingers remain stuck against the table while closing. On the other hand, the power drill is the heaviest object in the set (1190g). Heavy objects require more solid grasps to be lifted. Including force measurements into the planning algorithm to evaluate the robustness of a grasp might increase the grasping success for heavy objects.

Finally, we compare the proposed method with the one developed in [20], where MVBB decomposition is used as well, but the grasping poses are generated based on geometric considerations, and multiple variations of the generated grasps are tested in simulation. In [20], the efficacy of the method is tested in simulation with 4 objects perfectly reproduced by the point cloud. The average grasping success rate in [20] is 77.61%. In order to compare the two methods, we tested our approach on the object in [20] with the worst and best performance, i.e., the Mug (52.5%) and the Pot (97.5%). For the single item, we obtain 100% of success for the Mug and 80% success for the Pot (one failure over five trials was due to the poor quality of the point cloud and consequently of the box decomposition). The average for these two objects is 75% for [20] versus the 90% obtained by the novel approach.

## V. Conclusions

We proposed and validated a planning algorithm for grasping with robotic hands that encodes the expertise of a skilled user, trained in robotic hand use. The method starts with the acquisition of the point cloud of the object to be grasped that is approximated via cuboid bounding boxes. Then one box is selected and a DTR algorithm, based on grasps performed by the skilled human user on elementary boxes, generates a set of suitable hand poses to grasp this box. The grasps are ranked and checked for collision avoidance and kinematic feasibility. Finally, a candidate grasp is executed by the robot.

We tested the proposed approach using the Pisa/IIT SoftHand achieving a percentage of grasp success of 86.7% over 105 grasps on 21 previously unseen objects. Future work will focus on the integration of force measurements as possible indicators of the grasp quality. Segmentation algorithms could make the method also suitable for grasping grouped objects. Automatic selection of the parameter $t$ based on point cloud and hand parameters will be studied. Validation with different types of robotic hands will be carried out to explore the generality of the proposed method.

## References

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesisa survey. *T-Ro*, 30(2):289–309, 2013.

[2] A Bicchi. On the closure properties of robotic grasping. *IJRR*, 14(4):319–334, 1995.

[3] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[4] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *ICRA*, pages 1316–1322. IEEE, 2015.

[5] I. Lenz, Ho. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *IJRR*, 34(4-5):705–724, 2015.

[6] S. Kumra and C. Kanan. Robotic grasp detection using deep convolutional neural networks. In *IROS*, pages 769–776. IEEE, 2017.

[7] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: Computing robust robot vacuum suction grasp targets in point clouds using a new analytic model and deep learning. *arXiv preprint arXiv:1709.06670*, 2017.

[8] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[9] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. *arXiv preprint arXiv:1905.10520*, 2019.

[10] C Piazza, G Grioli, MG Catalano, and A Bicchi. A century of robotic hands. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:1–32, 2019.

[11] A. Rocchi and K. Hauser. A generic simulator for underactuated compliant hands. In *SIMPAR*, pages 37–42. IEEE, 2016.

[12] C Della Santina, C Piazza, G. M. Gasparri, M. Bonilla, M. G. Catalano, G. Grioli, M. Garabini, and A. Bicchi. The quest for natural machine motion: An open platform to fast-prototyping articulated soft robots. *RAM*, 24(1):48–56, 2017.

[13] R. Deimel and O. Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. *IJRR*, 35(1-3):161–185, 2016.

[14] C. Della Santina, V. Arapi, G. Averta, F. Damiani, G. Fiore, A. Settimi, M. G. Catalano, D. Bacciu, A. Bicchi, and M. Bianchi. Learning from humans how to grasp: a data-driven architecture for autonomous grasping with anthropomorphic soft hands. *RA-L*, 4(2):1533–1540, 2019.

[15] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. Template-based learning of grasp selection. In *IEEE ICRA*, pages 2379–2384, May 2012.

[16] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *IEEE ICRA*, pages 601–608, May 2013.

[17] G. Vezzani, U. Pattacini, and L. Natale. A grasping approach based on superquadric models. In *ICRA*, pages 1579–1586. IEEE, 2017.

[18] K. Huebner, S. Ruthotto, and D. Kragic. Minimum volume bounding box decomposition for shape approximation in robot grasping. In *ICRA*, pages 1628–1633. IEEE, 2008.

[19] A. Pas and R. Platt. Localizing handle-like grasp affordances in 3d point clouds. In *Experimental Robotics*. Springer, 2016.

[20] M. Bonilla, D. Resasco, M. Gabiccini, and A. Bicchi. Grasp planning with soft hands using bounding box object decomposition. In *IROS*, pages 518–523. IEEE, 2015.

[21] M. Bianchi, G. Averta, E. Battaglia, C.s Rosales, M. Bonilla, A. Tondo, M. Poggiani, G. Santaera, S. Ciotti, and M. G. et al. Catalano. Touch-based grasp primitives for soft hands: Applications to human-to-robot handover tasks and beyond. In *2018 ICRA*, pages 7794–7801. IEEE.

[22] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *RAM*, 22(3):36–52, 2015.

[23] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, pages 1957–1964. IEEE, 2016.

[24] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and Levin S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint:1806.10293*, 2018.