Vision-based grasp learning of an anthropomorphic hand-arm system in a synergy-based control framework

F. Ficuciello,[1*] A. Migliozzi,[1] G. Laudante,[2] P. Falco,[3] B. Siciliano,[1]

[1]Prisma Lab, University of Naples Federico II (DIETI), Naples, Italy
[2]Industry AMS, Naples, Italy
[3]ABB Corporate Research, Västerras, Sweden
[*]To whom correspondence should be addressed; E-mail: fanny.ficuciello@unina.it

In this work, the problem of grasping novel objects with an anthropomorphic hand-arm robotic system is considered. In particular, an algorithm for learning stable grasps of unknown objects has been developed, based on an object shape classification and on the extraction of some associated geometric features. Different concepts, coming from fields such as machine learning, computer vision and robot control have been integrated together in a modular framework in order to achieve a flexible solution suitable for different applications. The results presented in this work confirm that the combination of learning from demonstration and reinforcement learning can be an interesting solution for complex tasks, such as grasping with anthropomorphic hands. The imitation learning provides the robot with a good base to start the learning process that improves its abilities through trial-and-error. The learning process occurs in a reduced dimension subspace learned upstream from human observation during typical grasping tasks. Furthermore, the integration of a

synergy-based control module allows reducing the number of trials thanks to the synergistic approach.

Introduction

The ability to effectively manipulate different objects, and to successfully employ a variety of tools is one of the key skills that humans developed during their evolutionary history. For robots to approach the capabilities of humans to interact with their environment, the design, implementation and control of dexterous, anthropomorphic hands appear to be pivotal points, and have raised a lot of interest in the scientific community over the course of the years.

The human hand is a very complex, articulated biomechanical system, and the problem of replicating its structure and capability is very challenging, not only in terms of mechanical design, but also in terms of motion planning and control.

Inspired by studies in neuroscience (*1*), the idea of considering coordinated joint motion patterns for robotic hands, realized both by means of mechanical transmissions and by developing control strategies in a subspace of reduced dimensionality has been successfully applied in several works (*2*), (*3*), (*4*).

Many different approaches to grasp synthesis problem have been proposed, but most of them can be classified as either analytic or empirical (data-driven) (*5*). Analytic grasp synthesis usually consists of formulating a constrained optimization problem, in order to obtain grasps that satisfy one or more desired properties, such as stability and dexterity (*6*). Empirical approaches, instead, try to combine perceptual informations acquired through sensors and previous knowledge coming from either experience, or human demonstrations, in order to compute grasps that optimize some quality metrics (*7*). Different classifications have been proposed for data-driven algorithms: for example, in (*5*) the authors distinguish between techniques based on human observation, and those based on object features, while in (*7*) they are categorized based on the

amount of previous knowledge of the object (known, familiar, or unknown object). A key step in data-driven algorithms is grasp selection: based on the available data, an appropriate grasp can be selected, either by choosing one of the candidates in a database, or by synthesizing it from scratch. Generally the features of the object are fundamental in selecting a good grasp; different solutions have been proposed to associate grasps to different objects: in some works, the object is modeled using basic shape primitives, which are used to reduce the number of candidate grasps (*8*), (*9*), in (*10*) SVM (support vector machines) regression is used to match object shape, grasp parameters, and grasp quality, while in (*11*), (*12*) grasping regions of the object are identified.

After a grasp has been selected, it is usually evaluated, either in simulation, or on the real robot, according to some metrics that can discriminate between good and bad grasps. Over the years, a variety of metrics for evaluating grasp performances have been discussed. An extensive overview of the different quality indices proposed in literature is provided in (*13*). Grasp quality measures can be divided in two broad categories: those associated with the position of the contact points on the object, and those that depend on the hand configuration; the former can be further divided in three subgroups: measures based on the algebraic properties of the grasp matrix G, measures based on geometric considerations, and measures that keep into account limitations on the magnitude of the forces that can be exerted by the fingers.

Among Reinforcement Learning (RL) algorithms available in literature, it is worth to mention two of the model-based RL algorithms that can be suitable for robotic manipulation: PILCO (*14*) and PI-REM (*15*). PILCO (probabilistic inference for learning control) is a state-of-the-art model-based, policy search algorithm, introduced by M.P. Deisenroth and C.E. Rasmussen in 2011. PI-REM (Policy Improvement with REsidual Model) is a recent model-based algorithm proposed by M.Saveriano et al. in 2017. The key idea of PILCO is to learn a probabilistic forward model of the system dynamics, in order to explicitly take uncertainties into

account. The model is implemented as a non-parametric Gaussian Process with prior mean function and squared exponential kernel. The policy improvement is gradient-based, and the gradient is computed analytically. The basic idea of PI-REM is to develop initially an approximate model of the system, controlled with an approximate policy (learned with PILCO); in a second stage a residual model (difference between the approximate and real model) is learned and used to improve the real policy which is applied directly on the robot. In this work, we considered a robotic system consisting of a five-fingered hand and a redundant anthropomorphic manipulator, provided with many degrees of freedom (20 for the hand and 7 for the arm); therefore, we employed dimensionality reduction techniques in order to define the learning algorithm in a subspace of reduced dimensionality (*16*). In particular, the concept of postural synergies, originally introduced in (*1*) has been transfered to the robotic hand, as proposed in (*2*), (*17*), by means of principal component analysis (PCA). A data-driven approach to grasp synthesis, combining learning from demonstration based on Neural Networks, and reinforcement learning based on Policy Improvement with Path Integrals, is adopted to provide the robot with the ability to learn and improve grasps for previously unknown objects. To automatize the grasping process, a vision system is used to detect the object in the scene and to estimate its pose and geometric features. With respect to (*18*), the new contribution of this work consists in an upgraded framework that include visual perception and its integration in the learning pipeline. In (*18*) the geometric parameters of the objects are assumed to be known, while in this work they are computed from an RGB-D camera. Therefore, the level of autonomy of the algorithm has been increased. On the other hand, the accuracy and resolution of the camera as well as the efficiency of the algorithm adopted for geometry reconstruction and pose detection can affect the learning process. The experiments run in this work demonstrate that the algorithm is stable and robust in case of uncertainties on perception of the environment.

Results

Overview of the algorithm A schematic overview of the proposed algorithm is provided in Fig. 1.

First of all, informations about the region of interest in the scene are acquired in the form of a point cloud through an RGB-D sensor; the acquired data is filtered and processed by the *object recognition* module, that is responsible of detecting the objects in the scene, and estimating their shape, dimensions and pose.

The extracted features are sent as an input to the *Neural Networks* (NN) module, where two sets of multiple NN, one for the hand and one for the arm, have been trained on data acquired from human demonstration through motion capture tools; the provided output is a vector of three synergy parameters for the hand, and two coefficients for the arm. At this point, the parameters provided by the neural networks are used to initialize a reinforcement learning loop, based on the policy search paradigm, whose goal is to endow the robotic system with the ability to improve its performances over time, with reference to some appropriate quality metrics. Based on the initial policy parameters, some samples are extracted from Gaussian Multivariate Distributions for the coefficients of both the hand and the arm; each of these samples generates a trajectory for the robot, in Cartesian coordinates, during the planning phase. The planned trajectories are executed on the real system using a kinematic control strategy (*19*), where a closed loop inverse kinematics algorithm is used to convert references from the operative space to the configuration space of the robot, serving as inputs to the lower level controllers embedded in the system.

To each of the executed trajectories is assigned a cost that consists of two terms: a binary score that heavily penalizes trajectories leading to failed grasps, and a cost function related to a quality index of the force closure properties of the grasp (*20*). Finally, based on the obtained costs, the

policy is updated, and a new set of parameters is extracted; this loop can be either repeated for a fixed number of times, or run until the cost function becomes lower then a desired threshold.

Dimensionality reduction Robots are complex nonlinear systems, with many degrees of freedom, that have to execute complex actions, potentially interacting with unstructured environments. Therefore, most of the machine learning technique that have been successfully applied in other fields are difficult to apply to the huge amount of parameters involved in the context of robotics (*21*), (*22*).

A potential solution to this problem is to adopt different techniques that are targeted at reducing the dimensionality of the system. Postural synergies, sometimes also called eigengrasps, have been initially studied in the field of neuroscience: it has been observed that the movement of the human hand during grasping and manipulation is dominated by movements in a space of reduced dimensionality, compared to the number of DoFs of the human hand (*1*).

This concept can be transferred to robots by means of data analysis techniques, such as principal component analysis (PCA) (*23*). Taking advantage of this idea, the anthropomorphic hand can be controlled directly in the postural synergies subspace, greatly reducing the complexity of planning, control and learning. The postural synergies of the SCHUNK SVH hand computed in (*17*) have been implemented in this work. The number of motors is significantly lower than the number of joints, thus joint motion couplings are regulated by means of mechanical synergies defined via mechanical transmissions. The differential kinematics between the mechanical synergies subspace and the Cartesian space, is represented by the following equation $\dot{\boldsymbol{x}} = \boldsymbol{J}_{h_m}\dot{\boldsymbol{m}}$, where $\boldsymbol{J}_{h_m} \in \mathbb{R}^{n_x \times n_m}$ is the mechanical synergies Jacobian and is computed as $\boldsymbol{J}_{h_m} = \boldsymbol{J}_h\boldsymbol{S}_m$, such that,

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_h\boldsymbol{S}_m\dot{\boldsymbol{m}} = \boldsymbol{J}_h\dot{\boldsymbol{q}}; \tag{1}$$

$\boldsymbol{x} \in \mathbb{R}^{n_x}$, with $n_x = 15$, is the position vector of the five fingertips, $\boldsymbol{J}_h \in \mathbb{R}^{n_x \times n_q}$ is the S5FH hand Jacobian. $\boldsymbol{S}_m \in \mathbb{R}^{n_q \times n_m}$ is the matrix of the mechanical synergies and maps motor velocities into joint velocities, $\boldsymbol{S}_m \dot{\boldsymbol{m}} = \dot{\boldsymbol{q}}$. In addition to the mechanical synergies of the hand, motion coordination patterns or motor synergies can be computed to further reduce the number of parameters needed to plan and control the grasping activities. Synergies subspace for hand control has been computed using human grasp data and a mapping algorithm available from a previous work developed in (*24*). Human grasps data are based on fingertips measurements that are used as desired references in a closed-loop inverse kinematics (CLIK) scheme that is in charge of reconstructing the hand configuration. The maps between the synergies subspace and the motor space, and between the synergies subspace and the joint space are given respectively by $\boldsymbol{m} = \boldsymbol{S}_s \boldsymbol{\sigma} + \bar{\mathbf{m}}$, and $\boldsymbol{q} = \boldsymbol{S}_m(\boldsymbol{S}_s \boldsymbol{\sigma} + \bar{\mathbf{m}}) + \boldsymbol{q_0}$.

The matrix $\boldsymbol{S}_s \in \mathbb{R}^{n_m \times n_s}$ represents the base of the synergies subspace, whose dimensions depend on the number of eigengrasps considered to approximate the grasp. Thus, the number n_s can vary from 1 to 9. In this work, the synergies subspace is three-dimensional and, thus, it is obtained by choosing $n_s = 3$. Thus, the columns are the first three eigenvectors (or eigengrasps) with higher variance computed using principal component analysis (PCA) on the grasps data-set mapped from the human hand to the robotic hand. Finally, $\bar{\mathbf{m}} \in \mathcal{M} \subseteq \mathbb{R}^{\mathbf{n_m}}$ is the zero-offset of the motor synergies subspace.

Object recognition and pose estimation. In this work, a simple object recognition and pose estimation pipeline has been implemented.We used a semi-structured environment involving a table on which different objects are placed. To simplify, we assume that such objects could be recognized as sphere or cylinder. We used an ASUS Xtion PRO live camera for point cloud data acquisition.

We divide the analysis in two phases: syntactic phase, in which different clusters have been

extracted from the scene; semantic phase, in which all the clusters found are tested with two RANSAC models (sphere and cylinder) and the object is labeled as the best fitting model.

Syntactic phase. After taking a point cloud of the scene, we follow a simplified processing pipeline. Since we have both a static scene and static position between the scene and the camera, we can assume that all the data out of certain limits can be cut out and filtered. As a second step, we find the largest planar component of the scene, representing the table, and extract it; we also need to divide all the other points in different clusters so that we obtain all the clusters that represents all the candidate objects that will be classified (see Fig. 2). For this purpose, we implement the Euclidean cluster segmentation criteria described as follow:

- The point cloud P is represented in a k-d tree structure;
- For each point p_i in P:
  - Add p_i to the current queue Q;
  - For each point p_i in Q do:
    - add every neighbor of p_i in a radius r, to Q unless it has already been processed;
  - When the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to an empty list;
- The algorithm terminates when all points $p_i \in P$ have been processed and are now part of the list of point clusters C

In Algorithm 1 we describe in short how the syntactic phase work. The input to the syntactic phase algorithm is a point cloud S of the scene and the output is the vector of point cloud $clusters[pointCloud]$.

Semantic phase. Now we have to label the different objects found. Each object is processed both as cylinder and a sphere using RANSAC models. The criteria adopted is related to the

Algorithm 1 Syntactic Phase

1: function $(clusters[pointCloud]) = \text{syntacticPhase}(\text{PointCLoud}S)$
2: $(S_f) = \text{regionFilter}(S, regionLimits)$
3: $(S_f) = \text{removeLargestPlanarComponent}(S_f)$
4: $(clusters[pointCloud]) = \text{EuclideanClusterExtraction}(S_f)$

number of inliers and outliers from which the RANSAC algorithm comes out. In this way we can classify each object and establish how much it is sphere or cylinder. Both spheres and Cylinders are associated to a vector of parameters $\boldsymbol{\vartheta}$. For spheres it is $\boldsymbol{\vartheta} = (r, c)$, where r and c are the radius and the 3D center of the mathematical model respectively. For cylinders the parameter are $\boldsymbol{\vartheta} = (r, c, d, h)$ where r is the radius, c is the 3D centroid of the cluster, d is the vector representing the axis of the cylinder and h is the height computed as the distance between two point that have the maximum and minimum coordinate values on the maximum variance axis. Such criteria allows us to also use different objects that are quasi-sphere or quasi-cylinder (such as a cup of tea or a bottle). This approach emphasizes the robustness of the system to grasp or manipulate objects that have a similar shape to those already known. In fact, in the recognition phase we assume that all the objects can be sufficiently approximated with sphere or cylinders. Even though this approximation can introduce errors in recognition and pose estimation, the learning process makes the system robust to perception errors. The pseudocode of the semantic phase is described in Algorithm 2. The input to the semantic phase algorithm is a point cloud O of a clustered object and the outputs are the label l and the parameter vector $\boldsymbol{\vartheta}$ of the sphere or cylinder that best fits the object.

Pose estimation. RANSAC model has different parameters for each shape: center and radius for the sphere; center, radius, axis of the cylinder and height for the cylinder. All the coordinates are expressed in camera frame, thus a transformation in the robot frame is necessary to allow for manipulation. For this purpose, a marker has been placed at a fixed location on the base of the robot and transformation matrices have been used to move from one coordinate

Algorithm 2 Semantic Phase

function $(l, \boldsymbol{\vartheta}) = \text{semanticPhase}(\text{PointCLoud}O)$
$(i_s, \boldsymbol{\vartheta}_s) = \text{RANSAC}(O, Sphere)$
$(i_c, \boldsymbol{\vartheta}_c) = \text{RANSAC}(O, Cylinder)$
if $i_c > i_s$ **then**
    $l = Cylinder$
    return $(l, \boldsymbol{\vartheta}_c)$
else
    $l = Sphere$
    return $(l, \boldsymbol{\vartheta}_s)$
end if

frame to another.

The following notations for the necessary transformation matrices can now be defined.

- $\boldsymbol{T_o^c}$: Pose of the object in camera frame;
- $\boldsymbol{T_c^m}$: Transformation between the camera and the marker;
- $\boldsymbol{T_m^b}$: Constant transformation between the marker and the base frame;
- $\boldsymbol{T_o^b}$: Transformation between the object and the base frame;

$\boldsymbol{T_o^c}$ is provided by the object recognition script, $\boldsymbol{T_c^b}$ is acquired using a ROS wrapper for *Alvar* (*25*), an open source AR tag tracking library, while $\boldsymbol{T_m^b}$ is fixed and known.
For the composition property of transformation matrices (*19*), the desired matrix $\boldsymbol{T_o^b}$ can be found as:

$$\boldsymbol{T_o^b} = \boldsymbol{T_m^b T_c^m T_o^c} \tag{2}$$

Learning from demonstration In a previous work (*26*), a dataset of grasps demonstrated by a human teacher has been acquired through a motion capture suit. For each of the demonstrated grasp, the features of the object have been defined as the input, while the values of both the

synergy coefficients of the hand, and the parameters of the arm have been defined as the output. This dataset has been used to train 2 sets (one for the hand and one for the arm) of neural networks (*27*). Their architecture is experimentally chosen by trying different combinations of hidden layers and neurons and analyzing the corresponding performance in terms of Mean Squared Error (MSE). We found out that a feedforward structure with 2 hidden layers and 10 neurons for each is the suitable choice for the particular application.

Reinforcement learning The output of the NN module is then used to initialize the policy parameters of a reinforcement learning loop. The idea is to explore the synergy subspace of the system locally to the initial values provided by imitation learning, in order to improve the performances of the robot in a trial-and-error paradigm. The reinforcement learning loop can be fundamentally divided in 4 main phases:

- **Sampling:** N samples are extracted from two Multivariate Gaussian Distributions with means $\boldsymbol{\mu}_{hand}$ and $\boldsymbol{\mu}_{arm}$ given by the outputs of the neural networks, and covariance matrices $\boldsymbol{\Sigma}_{hand}$ and $\boldsymbol{\Sigma}_{arm}$ chosen before the start of the loop. The choice of $\boldsymbol{\Sigma}_{hand}$ and $\boldsymbol{\Sigma}_{arm}$ determines the width of the exploration in the space of policy parameters; in this work, a simple exploration decay has been implemented in order to favor the exploitation of the acquired informations, over exploration in later stages of the learning process.

- **Planning:** For each of the samples extracted in the previous phase, a trajectory in the Cartesian space is planned for both the hand and the arm; the equation mapping the synergy subspace to the Cartesian space for the hand is given by:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_{hm}\dot{\boldsymbol{m}} = \boldsymbol{J}_{h}\boldsymbol{S}_{s}\dot{\boldsymbol{\sigma}} \tag{3}$$

where $\boldsymbol{J}_{hm}$ is the mechanical synergies Jacobian, $\dot{\boldsymbol{m}}$ and $\dot{\boldsymbol{\sigma}}$ are the vectors of motor and

synergies velocities respectively, $\boldsymbol{J}_h$ is the hand Jacobian, and $\boldsymbol{S}_s$ is the synergy matrix; instead, the mapping between the reduced subspace for the arm and the Cartesian pose of the end-effector is given by:

$$\boldsymbol{p} = \overline{\boldsymbol{p}} + \boldsymbol{e}_p \alpha_p \tag{4}$$

$$\boldsymbol{\epsilon} = \overline{\boldsymbol{\epsilon}} + \boldsymbol{e}_\epsilon \alpha_\epsilon \tag{5}$$

where $\boldsymbol{p}$ is the position of the center of the wrist, $\boldsymbol{\epsilon}$ is the vector part of the quaternion, $\overline{\boldsymbol{p}}$ and $\overline{\boldsymbol{\epsilon}}$ are the mean values of $\boldsymbol{p}$ and $\boldsymbol{\epsilon}$ obtained from PCA and $\boldsymbol{e}_p$ and $\boldsymbol{e}_\epsilon$ are the first principal component for the position and orientation respectively. Based on this relationships, a rectilinear path from the home configuration of the wrist, to the desired pose is planned; once the desired pose has been reached, the fingers are closed according to the values of the synergy coefficients.

- **Execution:** In order to execute the planned trajectories, the references have to be mapped in the configuration space of the system; in particular, we need to compute the target velocities of the joints that allow the robot to reproduce the desired motions: this has been done by applying a closed loop inverse kinematic algorithm (*19*).

  In addition to the execution of the planned motions for the hand-arm system, an additional, low-level reactive control has been implemented for the hand: the fingers close towards the centroid of the polygon with vertices at the center of the fingertips until the measured current reach a certain threshold.

- **Evaluation and reward**: For each trial performed the agent receives a reward function. As in (*28*), the scalar cost function $V(\boldsymbol{\sigma})$ is based on a force-closure quality index in-

troduced in (*20*). Without going into details, an algorithm for optimal force distribution towards the improvement of force closure property can be based on the minimization of a cost function $V(\delta\boldsymbol{\sigma})$ with respect to $\delta\boldsymbol{\sigma}$.

Let $\Omega_{i,j}^{k} \subset \mathbb{R}^{h}$ indicate the set of grasp variables $\boldsymbol{y}$ that satisfy the friction cone constraint with a (small, positive) margin k, where h is the dimension of the internal force subspace $\boldsymbol{E}$; and let $\boldsymbol{w}$ indicates the object weight. The matrix $\boldsymbol{E}$ maps the controlled joint displacement into internal forces activated on the object. Underactuation reduce the subspace's dimension of controllable internal forces according to the mechanical and motor synergies and, thus, according to the matrices $\boldsymbol{E}_m = \boldsymbol{E}\boldsymbol{S}_m$ and $\boldsymbol{E}_s = \boldsymbol{E}\boldsymbol{S}_m\boldsymbol{S}_s$.

For the $i-th$ contact and the $j-th$ constraint, V is obtained as the summation of the terms: $V(\boldsymbol{w},\boldsymbol{y}) = \sum_i \sum_j V_{i,j}(\boldsymbol{w},\boldsymbol{y})$, defined as

$$V_{i,j} = \begin{cases} (2\sigma_{i,j}^{2}(\boldsymbol{w},\boldsymbol{y}))^{-1} & \boldsymbol{y} \in \Omega_{i,j}^{k} \\ a\sigma_{i,j}^{2}(\boldsymbol{w},\boldsymbol{y}) + b\sigma_{i,j}(\boldsymbol{w},\boldsymbol{y}) + c & \boldsymbol{y} \notin \Omega_{i,j}^{k} \end{cases} \tag{6}$$

where a, b and c are constant positive parameters conditioned by properties imposed to V.

This cost function, detailed in (*20*), has a formulation, suitable for practical implementation in manipulation planning. To be more specific, the function V has been adopted as a cost function indicating the quality of grasp since the reciprocal of V reflects the distance of the grasp from violating the friction cone constraints and is obtained by formulating and solving the problem as a second order cone programming (SOCP).

In this work, the reward function $r\left(\boldsymbol{\sigma}\right)$ has the following expression:

$$r\left(\boldsymbol{\sigma}\right) = \beta V\left(\boldsymbol{\sigma}\right) + \phi \tag{7}$$

where $\beta = 10^{-6}$ is a scaling factor and ϕ is:

$$\phi = \begin{cases} 0 & \text{if grasp succeeds} \\ 10^{4} & \text{if grasp fails} \end{cases} \tag{8}$$

The value $\phi = 10^4$ has been chosen higher than in the previous work in order to penalize more decisively the failed grasps and to avoid them in the following explorations. β and ϕ have been tuned experimentally with the aim that the cost function $V(\boldsymbol{\sigma})$ become meaningful when the grasp is successful.

Experimental results We tested our algorithm on an experimental setup, consisting of a Schunk five fingered hand, a Kuka lightweight robot 4+, an Asus Xtion Pro Live sensor for point cloud acquisition and a PC equipped with the ROS meta-operating system.

For each experiment a different object is chosen, and placed at a random position on a table in front of the robot; the sensor acquires a point cloud of the scene, and the features and pose of the object are estimated by the object recognition module, which publishes these informations on a ROS topic, that the neural network node subscribes to. The initial values of the policy parameters are obtained by the NN and used to generate 5 samples for the reinforcement learning loop; each of the corresponding trajectories is executed and a reward is computed based on the force closure cost function, and on the result of a simple lifting test: once the fingers are closed, the robot tries to lift the object and carry it at the home configuration: if the test fails, an high penalty is assigned to the trial, while obviously no penalty is assigned for a successful test. The rewards associated to each trajectory are used to update the policy of the algorithm, assigning low probabilities to trajectories that performed poorly.

Three different objects have been considered for these experiments: a tennis ball, a plastic strawberry, and a plastic bottle; the results of the experiments are shown in Figures 3, 4, and 5. For each experiment we report:

- a color-coded table, were the i-th row is the i-th update of the reinforcement learning loop, and the j-th column is the j-th trial: each entry is red if the corresponding grasp failed, green if it was successful. Thus, five trials for each update led to a total number of 25

trials. For each tried grasp, the matching with the corresponding values on the graphs (images **(B)**, **(C)** and **(D)**) can be found by reading the table from left to right and from the top to the bottom;

- a plot of the evolution of the force closure cost function;
- a plot of the evolution of the hand synergies and arm scores.

We can see how the number of successful grasps significantly increases during the final updates in all the experiments, while the cost associated with the force closure index converges to lower values; in particular, in the experiment with the tennis ball, we can see how the number of successful attempts is higher in the beginning of the experiment compared with the other objects: in fact, since the ball can be accurately represented with a simple spherical shape, the values of the parameters obtained by imitation learning already provide a good approximation for synthesizing a stable grasp; instead, when trying to execute harder grasps, the informations acquired from human demonstration are not good enough to obtain stable grasps, and the reinforcement learning loop is necessary to explore the parameter space and improve the initial grasps by learning from the interaction of the robot with the environment.

The role of force closure cost function In order to appreciate the influence of the two terms of the reward function in the convergence of the algorithm, we have run experiments (using the value $\phi = 10^3$) for: (i) the whole hand-arm system; (ii) only for the hand. The conclusion is that for the hand-arm system we have to increased ϕ, as done in this work ($\phi = 10^4$), to have a faster convergence (almost halving the trials). Indeed, by increasing ϕ, the pose of the arm leading to a correct grasp is learned faster. The experiments confirm that the learning capabilities and the convergence of the algorithm are significantly affected by the parameter ϕ while the force closure cost function affect mainly the hand configuration. In order to appreciate

the influence of the force closure cost function, we have run the algorithm with and without the adoption of this cost only for the hand. The conclusion is that the force closure cost function does not influence the success of the grasp but the stability of the grasp. The number of trials to get the convergence of the algorithm does not change while the force closure cost changes significantly. In Table 1 a quantitative comparison of final grasps, obtained for different objects with and without the adoption of the quality index to generate the action, is reported.

Table 1: Comparison of grasps obtained with and without force closure cost function.

Force Closure Cost Value		
Grasp	Cost not used	Cost used
Bipodal	$1.9 \cdot 10^7$	$8.5 \cdot 10^5$
Tripodal	$2.5 \cdot 10^7$	$3.1 \cdot 10^6$
Sphere five finger	$2.7 \cdot 10^8$	$4.6 \cdot 10^7$
Cylinder five finger	$1.9 \cdot 10^8$	$1.4 \cdot 10^7$

Evaluation of the algorithm in different initial conditions To validate the algorithm in different initial conditions we have compared the results using force closure cost function after 25 trials. We have tested the algorithm in three different conditions with reference to the parameters initialization: (i) the initialization of the policy is provided by comparing the object to be grasped with examples contained in a reference table (including a limited number of object/grasp pairs) and by choosing parameters related to the closest one; (ii) the shape and dimension of the object are known and given as input to the NN, the initialization of the policy is the output of the NN; (iii) the object is unknown and the vision system is in charge of extracting the information to be given as input to the NN. In case (i) the algorithm fails and does not converge. In (ii) and (iii) the convergence is ensured but the quality of the grasp is influenced by the uncertainty introduced by the perception. The results are better for (ii) in terms of quality of the grasp. For (iii) the cost function is greater of an order of magnitude, regardless object and grasp type. Therefore, we can state that the algorithm is robust to uncertainties on perception

to a certain extent (see cases (ii) and (iii)), but if the initialization is too far from the correct value the algorithm does not converge (see case (i)). Another consideration to do is that if the system is constituted only by the hand, in case (i) the algorithm converge. This means that the robustness of the algorithm with respect to the initial conditions decreases as the complexity of the system increases.

Discussion

In this work, the problem of grasping novel objects with a robotic hand-arm system has been considered; in particular, an algorithm for learning stable grasps of unknown objects has been developed, based on a simple shape classification and on the extraction of some of the associated features.

Building upon previous researches on the topic, the results presented in this work confirmed that the combination of learning from demonstration and reinforcement learning can be an interesting solution for complex tasks, such as grasping with anthropomorphic hands, providing the robot with a good base to start the learning process, and allowing it to improve its abilities through trial-and-error.

The experiments carried out on our setup provided encouraging results, showing how this framework is suitable for synthesizing stable grasps of different objects, and how they can potentially be improved over time, with reference to some quality metrics.

The algorithm has been tested in three different conditions with reference to parameters initialization. The experimental results demonstrate that the system is robust to uncertainties on perception of the environment to a certain extent. If the initialization is ”wrong”, namely too far from a good value, the algorithm does not converge. The initialization is obtained using a neural network trained by human grasping samples. We have verified that, if neural networks are not used to initialize the policy, the algorithm does not converge. In other words, a complex

system with high degrees of freedom would never converge unless proper initialization of the policy parameters. In this contest synergies will help for a smart choice of the policy while a synergy-based supervised learning algorithm allows a suitable initialization of the policy parameters. One of the main limitations of the proposed algorithm is in the very simple object recognition algorithm, which could be improved by considering additional shape templates, as well as decomposing complex objects into simpler parts that can be associated with basic shapes in a satisfying way; furthermore, an automatic way for testing the success of a grasp, without relying on a supervisor could be implemented: for example this could be done by tracking the object position with the camera.

The use of different cost functions in the reinforcement learning loop could also be investigated, for example developing some metric for the quality of a grasp derived from visual informations, or considering costs that depend on the particular task that we want to accomplish.

The proposed algorithm could also be integrated in a wider framework, where different actions are selected on an higher hierarchical level in a task-oriented fashion, guiding the learning process towards grasp that better suit a particular application. Future comparison with different learning algorithm are referred in particular to reinforcement learning policy search methods like PILCO (Probabilistic Inference for Learning Control) (*14*) and PI-REM (Policy Improvement with REsidual Model) (*15*). It is worth noticing that, since it is not easy to integrate these algorithms in such a complex framework, the comparison with other methods is not trivial. In effect, different aspects need to be considered and carefully evaluated.

Materials and methods

Hardware

Schunk five-finger hand (SVH) The SVH is a very compact, compliant under-actuated hand, which closely resembles the structure and the appearance of the human hand; one of the key features of this hand, is that all controllers, regulators, and power electronics are fully integrated in the wrist. The hand has a total of 9 drives that move the 20 degrees of freedom of the hand, thanks to the mechanical coupling between joints, inspired by the way human fingers usually move together. Technical details and specifics for this device can be found in (*29*). In order to communicate with the hand, a ROS wrapper for the SVH driver, available at (*30*) has been used.

Kuka Lightweight Robot 4+ The KUKA LWR 4+ is a very efficient and portable robot, weighting only 16 kg, with a payload of 7 kg; motors, gears and sensors are accommodated inside an aluminum housing, as well as the necessary control and power electronics.
The redundant DoF provides the arm with additional flexibility, and can be used in different ways, based on the task at hand: it can be exploited to avoid obstacle, to increase the manipulability, or in general to obtain more favorable motions for the desired task.
Technical details and specifics for this device can be found in (*31*).
The control of the KUKA arm has been implemented by taking advantage of the FRI library, available at (*32*), which provides a simple user interface for communicating with the robot. The library runs on a remote pc, that is connected with the KRC (KUKA Robot Controller) via ethernet.

Asus Xtion Pro Live Asus Xtion PRO LIVE (*33*) is an RGB-D sensor, consisting of an RGB camera, an infrared emitter, and a CMOS sensor. The Asus Xtion PRO LIVE is a compact, plug-and-play device, with a resolution of 640x480 for the depth stream, and 1280x1024 for the color stream, with a depth range of approximately 0.6 to 3.5 m and a field of view of 58 horizontal, 45 vertical.

Software

ROS ROS is a meta-operating system, providing many functionalities, such as low-level device control, hardware abstraction, and package management. ROS is based on a peer-to-peer network of processes, called nodes, which communicates using the ROS communication infrastructure.
The ROS framework is based on an asynchronous publish/subscribe system, and it provides many different functionalities, easily accessible, either by GUI, or by command-line (*34*).

PCL Library The PCL library provides implementations of many state-of-the-art algorithms for point cloud processing, such as filtering, segmentation and model fitting (*35*).

Methods

Neural networks The NN used in this work have been built and trained using the *Neural Networks Toolbox* available in $Matlab^{©}$. All the networks have the same architecture, chosen experimentally, consisting of two hidden layers with 5 neurons each; the available dataset has been randomly divided in training, validation and test set, with 70%, 15% and 15% percentages respectively. The weights have been randomly initialized using the Nguyen-Widrow rule, and the Levenberg-Marquadt algorithm has been used for training.

Policy improvement with path integrals (PI2) The implemented solution for the reinforcement learning algorithm loop is based on the episodic PI2 formulation originally proposed in (*36*). Each trajectory is assigned a probability in the following way:

$$P(\tau_i) = \frac{e^{-\frac{1}{\lambda}S_i}}{\sum_{i=1}^{n} e^{-\frac{1}{\lambda}S_i}} \tag{9}$$

where $S_i = S(\tau_i)$ is the cost of trajectory τ_i.

These probabilities are used to update the mean value of the parameters for the next iteration:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}k} = \boldsymbol{\mu}_{\boldsymbol{\theta}k-1} + \sum_{i=1}^{n} P(\tau_i)(\boldsymbol{\mu}_{\boldsymbol{\theta}i-1} - \boldsymbol{\theta}_i) \tag{10}$$

where

- $\boldsymbol{\mu}_{\boldsymbol{\theta}0}$: initial mean value of the policy parameters;
- $\boldsymbol{\Sigma}_{\theta 0}$: initial covariance matrix
- $\boldsymbol{\theta}_i \sim N(\boldsymbol{\mu}_{\theta 0}, \boldsymbol{\Sigma}_{\theta 0})$: i-th policy parameters sample;

Since we are implementing an exploration decay, $\boldsymbol{\Sigma}_\theta$ is also updated:

$$\boldsymbol{\Sigma}_{\theta k} = \gamma \boldsymbol{\Sigma}_{\theta k-1} \tag{11}$$

The values of the parameters used for the experiments are:

- $k = 5$: number of updates
- $n = 5$: number of trials per update
- $\lambda_{hand} = 1000$: hand synergies variance
- $\lambda_{arm} = 0.0002$: arm coefficients variance
- $\gamma = 0.7$: exploration decay

Figures

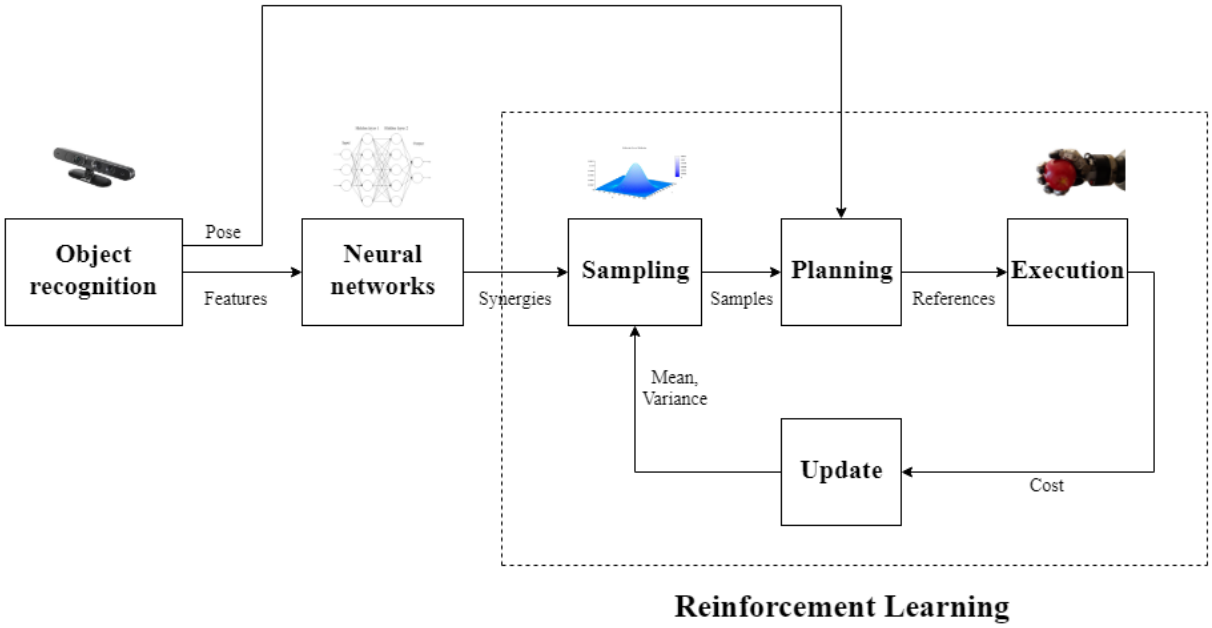

Figure 1: Schematic overview of the proposed algorithm.

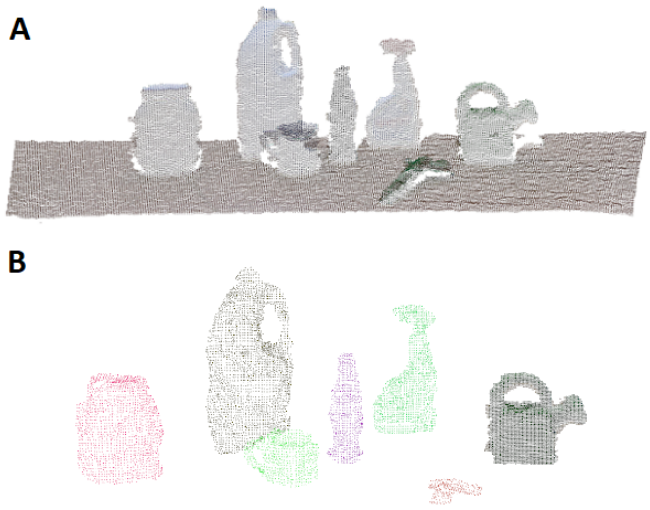

Figure 2: Example of point cloud processing: **(A)** Original point cloud, **(B)** Processed point cloud: the main plane in the scene has been removed, and segmentation of the remainder of the cloud has been executed.

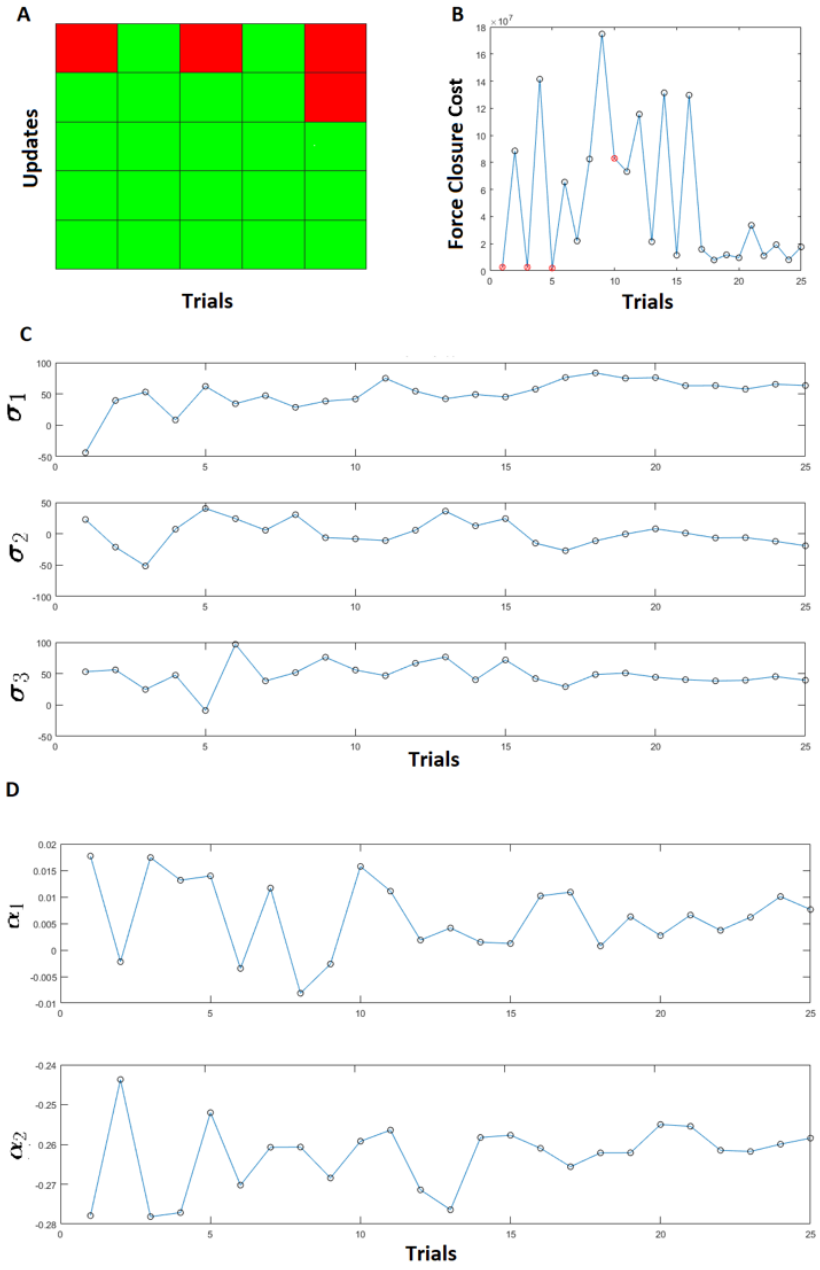

Figure 3: Experiment on a tennis ball: **(A)** Grasp success table, **(B)** Force closure cost function, **(C)** Synergy coefficients, **(D)** Arm scores.

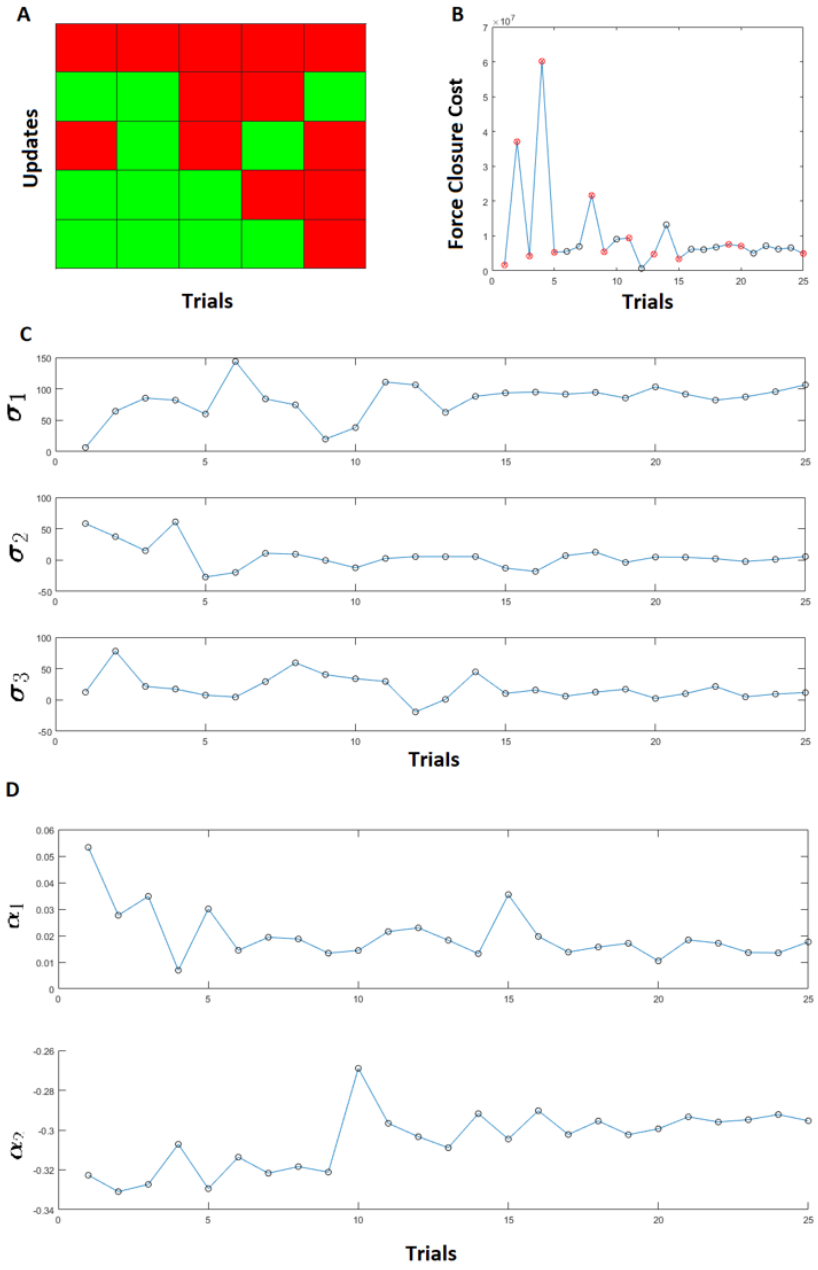

Figure 4: Experiment on a plastic strawberry: **(A)** Grasp success table, **(B)** Force closure cost function, **(C)** Synergy coefficients, **(D)** Arm scores.

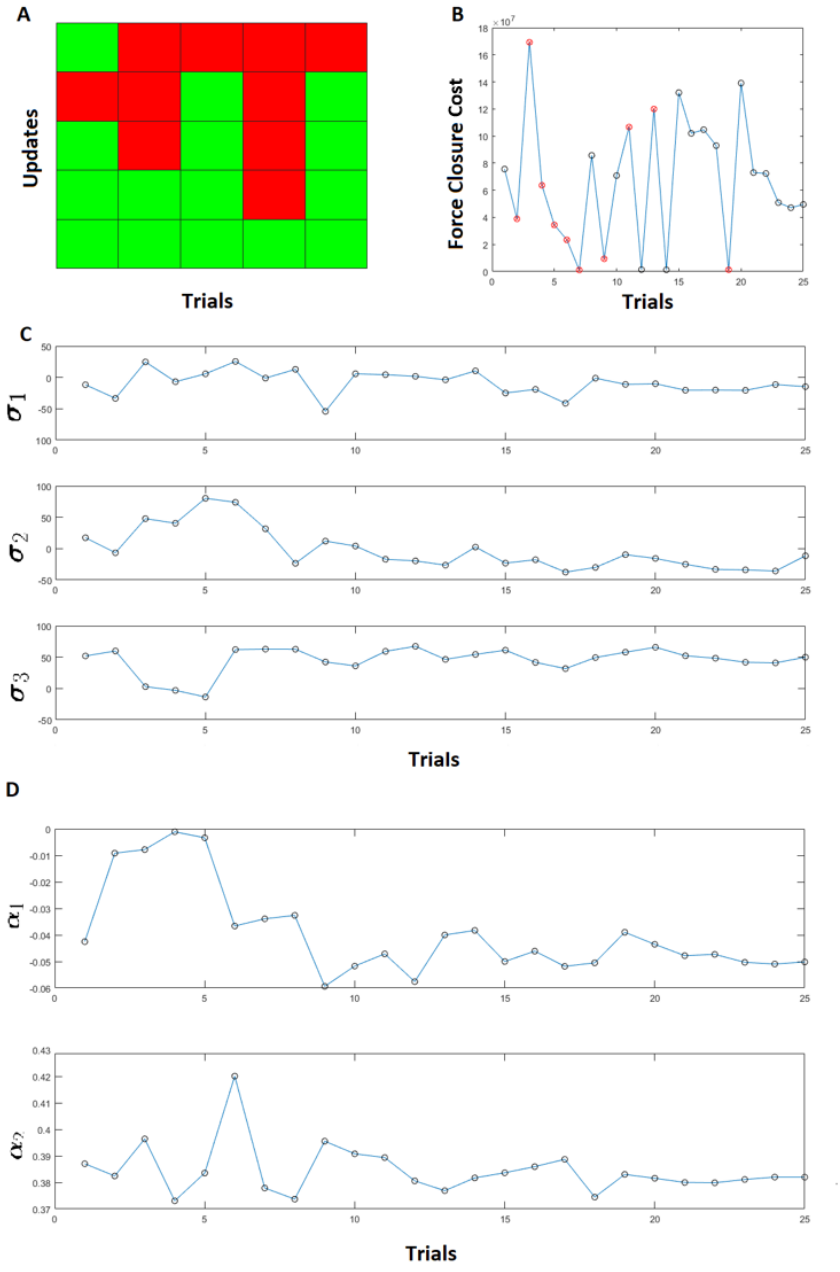

Figure 5: Experiment on a plastic bottle: **(A)** Grasp success table, **(B)** Force closure cost function, **(C)** Synergy coefficients, **(D)** Arm scores.

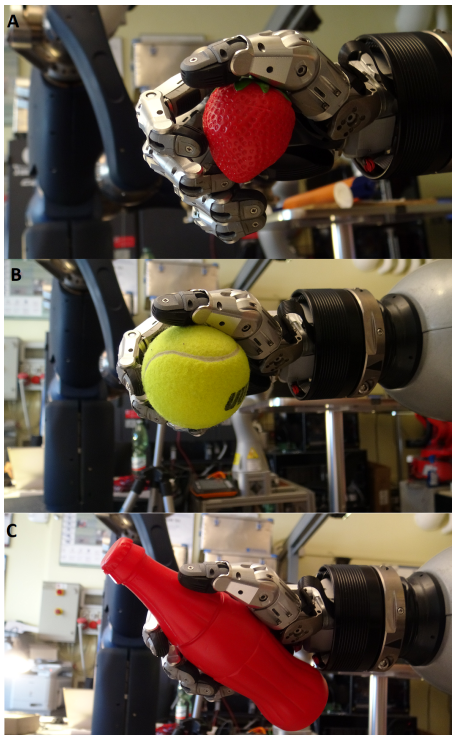

Figure 6: Successful grasp of the tennis ball, strawberry and plastic bottle at the end of the learning process.

References and notes

References

1. M. Santello, M. Flanders, J. F. Soechting, Postural hand synergies for tool use, *Journal of Neuroscience* **18**, 10105 (1998).

2. M. Ciocarlie, P. Allen, Hand posture subspaces for dexterous robotic grasping, *The International Journal of Robotics Research* **28**, 851 (2009).

3. M. Ciocarlie, C. Goldfeder, P. Allen, Dimensionality reduction for hand-independent dexterous robotic grasping, *IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 3270–3275 (2007).

4. M. Gabiccini, A. Bicchi, D. Prattichizzo, M. Malvezzi, On the role of hand synergies in the optimal choice of grasping forces, *Autonomous Robots* **31**, 235 (2011).

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

6. K. B. Shimoga, A. A. Goldenberg, Soft robotic fingertips: Part i: A comparison of construction materials, *The International Journal of Robotics Research* **15**, 320 (1996).

7. J. Bohg, A. Morales, T. Asfour, D. Kragic, Data-driven grasp synthesisa survey, *IEEE Transactions on Robotics* **30**, 289 (2014).

8. A. T. Miller, S. Knoop, H. I. Christensen, P. K. Allen, Automatic grasp planning using shape primitives, *IEEE International Conference on Robotics and Automation* pp. 1824–1829 (2003).

9. K. Huebner, D. Kragic, Grasping by parts: Robot grasp generation from 3d box primitives, *4th International Conference on Cognitive Systems* (2010).

10. R. Pelossof, A. Miller, P. Allen, T. Jebara, An svm learning approach to robotic grasping, *IEEE International Conference on Robotics and Automation* pp. 3512–3518 (2004).

11. A. Saxena, J. Driemeyer, A. Y. Ng, Robotic grasping of novel objects using vision, *The International Journal of Robotics Research* **27**, 157 (2008).

12. M. Stark, P. Lies, M. Zillich, J. Wyatt, B. Schiele, Functional object class detection based on learned affordance cues, *International conference on computer vision systems* pp. 435–444 (2008).

13. M. A. Roa, R. Suárez, Grasp quality measures: review and performance, *Autonomous robots* **38**, 65 (2015).

14. M. Deisenroth, C. Rasmussen, *In Proceedings of the 28th International Conference on machine learning* (2011), pp. 465–472.

15. M. Saveriano, Y. Yin, P. Falco, D. Lee, Data-efficient control policy search using residual dynamics learning, *IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 4709–4715 (2017).

16. F. Ficuciello, P. Falco, S. Calinon, A brief survey on the role of dimensionality reduction in manipulation learning and control, *IEEE Robotics and Automation Letters* **3**, 2608 (2018).

17. F. Ficuciello, A. Federico, V. Lippiello, B. Siciliano, Synergies evaluation of the schunk s5fh for grasping control, *Advances in Robot Kinematics 2016* pp. 225–233 (2018).

18. F. Ficuciello, Hand-arm autonomous grasping: Synergistic motions to enhance the learning process, *Intelligent Service Robotics* (2018).

19. B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics–modelling, planning and control, *Advanced Textbooks in Control and Signal Processing Series* (2009).

20. A. Bicchi, On the closure properties of robotic grasping, *The International Journal of Robotics Research* **14**, 319 (1995).

21. J. Peters, S. Vijayakumar, S. Schaal, Reinforcement learning for humanoid robotics, *IEEE-RAS international conference on humanoid robots* pp. 1–20 (2003).

22. M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, A survey on policy search for robotics, *Foundations and Trends® in Robotics* **2**, 1 (2013).

23. F. Ficuciello, G. Palli, C. Melchiorri, B. Siciliano, Postural synergies of the ub hand iv for human-like grasping, *Robotics and Autonomous Systems* **62**, 515 (2014).

24. G. Palli, C. Melchiorri, G. Vassura, U. S. et al., The dexmart hand: Mechatronic design and experimental evaluation of synergy-based control for human-like grasping, *Int Journal of Robotics Research* **33**, 799 (2014).

25. ROS, Ros wrapper for the alvar AR tag tracking library. `http://wiki.ros.org/ar_track_alvar`.

26. F. Ficuciello, D. Zaccara, B. Siciliano, Learning grasps in a synergy-based framework, *International Symposium on Experimental Robotics* pp. 125–135 (2016).

27. M. P. Perrone, L. N. Cooper, *How We Learn; How We Remember: Toward an Understanding of Brain and Neural Systems: Selected Papers of Leon N Cooper* (World Scientific, 1995), pp. 342–358.

28. F. Ficuciello, D. Zaccara, B. Siciliano, Synergy-based policy improvement with path integrals for anthropomorphic hands, *IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 1940–1945 (2016).

29. Schunk, Svh documentation. `https://schunk.com/it_en/gripping-systems/highlights/svh/`.

30. Schunk, Svh driver ros wrapper. `http://wiki.ros.org/schunk_svh_driver`.

31. Kuka, Lightweight robot 4+ documentation. `https://www.kukakore.com/wp-content/uploads/2012/07/KUKA_LBR4plus_ENLISCH.pdf`.

32. FRILibrary, Fast research interface library. `https://cs.stanford.edu/people/tkr/fri/html/`.

33. Asus, Xtion pro live. `https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/`.

34. ROS, Robot operating system. `http://www.ros.org/`.

35. R. B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), *IEEE International Conference on Robotics and Automation* (2011).

36. F. Stulp, O. Sigaud, Path integral policy improvement with covariance matrix adaptation, *arXiv preprint arXiv:1206.4621* (2012).

Acknowledgments

The research leading to these results has been partially supported by the RoDyMan project (FP7/2007-2013) under ERC AdG-320992, and and partially by MUSHA project carried out in the frame of Programme STAR, financially supported by UNINA and Compagnia di San Paolo.