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

Cleaning tasks knowledge transfer between heterogeneous robots: a deep learning approach

Jaeseok Kim[1] · **Nino Cauli**[2] · **Pedro Vicente**[2] · **Bruno Damas**[2,3] · **Alexandre Bernardino**[2] · **José Santos-Victor**[2] · **Filippo Cavallo**[1]

Received: date / Accepted: date

Abstract Nowadays, autonomous service robots are becoming an important topic in robotic research. Differently from typical industrial scenarios, with highly controlled environments, service robots must show an additional robustness to task perturbations and changes in the characteristics of their sensory feedback. In this paper, a robot is taught to perform two different cleaning tasks over a table, using a learning from demonstration paradigm. However, differently from other approaches, a convolutional neural network is used to generalize the demonstrations to different, not yet seen dirt or stain patterns on the same table using only visual feedback, and to perform cleaning movements accordingly. Robustness to robot posture and illumination changes is achieved using data augmentation techniques and camera images transformation. This robustness allows the transfer of knowledge regarding execution of cleaning tasks between heterogeneous robots operating in different environmental settings. To demonstrate the viability of the proposed approach, a network trained in Lisbon to perform cleaning tasks, using the iCub robot, is successfully employed by the DoRo robot in Peccioli, Italy.

Parts of this manuscript were previously presented at the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC 2018), Torres Vedras

J.Kim · F.Cavallo
[1]BioRobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy.
E-mail: j.kim@sssup.it, filippo.cavallo@santannapisa.it

N.Cauli · P.Vicente · A.Bernardino · J.Santos-Victor
[2]Institute for Systems and Robotics, Instituto Superior Tecnico, Universidade de Lisboa, Portugal.
E-mail: {ncauli,pvicente,alex,jasv}@isr.tecnico.ulisboa.pt

B.Damas
[2]Institute for Systems and Robotics, Instituto Superior Tecnico, Universidade de Lisboa, Portugal.
[3]CINAV — Centro de Investigação Naval, Almada, Portugal.
E-mail: bdamas@isr.tecnico.ulisboa.pt

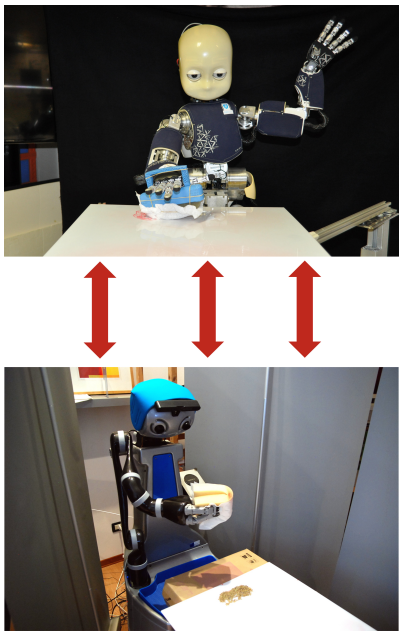


Fig. 1: Picture of the Lisbon iCub robot (top) and the Peccioli DoRo robot (bottom) in their experimental setup. Our system was trained on the iCub and then tested on the DoRo.

Keywords Learning from demonstration · Transfer learning · Data augmentation · Convolutional neural networks · Task parametrized Gaussian mixture models

1 Introduction

Times where robots were relegated to controlled factory environments, with absolutely no interaction with hu-

mans, are becoming part of our past. Nowadays, robots share their working environment with us, needing the ability to handle unexpected situations, to interact with humans, and to not interfering with co-workers actions (both humans and robotic). A perfect example of robotic platforms facing these problems are the service robots. In recent years, elderly population increased exponentially around the globe, forcing the research community to find solutions to smoothly integrate senior citizens in the modern society. During assistance, caregivers are overloaded with tasks, most of them physical and repetitive. For this reason, caregivers spend a substantial amount of their time in house chores and physical assistance, overlooking social interaction with the assisted elders. Service robots able to perform house chores would relieve caregivers from a significant burden, giving them more time to spend with the elders. Cakmak *et al.* [2] observed that cleaning tasks are 49.8% of all chores that humans perform at home. Many mobile cleaning robots were already successfully presented in the market, but they are able to perform only highly specialized and simplified tasks, like cleaning the floor inside an apartment with a predefined behaviour. Unfortunately, typical cleaning actions, such as wiping, washing, sweeping and scrubbing, require a robot with fine manipulation abilities to be performed [23]. As a result, several researchers focus their attention on service robots equipped with manipulators to perform cleaning tasks [32,42,25,15,31,9,33,41,22,24,21,14,27,7,8, 26,35,19,11,10,37,40,1,16,39,18,38].

The most direct approach to design a service robot able to perform basic cleaning actions is using classical control. Okada *et al.* [32] generate a sequence of body posture to perform a sweeping motion using whole body inverse kinematics. To increase stability and avoid self collisions, Yamazaki *et al.* [42] use the SR-inverse method to control robot's upper body during a cleaning task. Liang *et al.* [25] generate a sweeping motion with both arms using full dual position control based on task-space kinematics. Hess *et al.* proposed a novel coverage path planning for robotic manipulators that can clean arbitrary 3D surfaces [15]. The authors suggest a generalization of the traveling salesman problem (GTSP), which transforms the surface into a graph defining a set of clusters over nodes and minimizing some cost measures. Dornhege *et al.* discussed how to combine classical symbolic planning with geometric reasoning in their TFD/M (Temporal Fast Downward/Modules) planner for wiping tasks using the PR2 robot [31,9]. Ortenz *et al.* suggested projected operational space dynamics that minimize joint torque and increase stability while the robot is in contact with a whiteboard during a wiping movement [33]. Urbanek *et al.* used Cartesian impedance control to create a compliant behavior of the robotic end-effector while wiping a table [41]. The Cartesian impedance control is extended with a compliant whole-body impedance control framework to interact with the environment using Rollin' Justin in Leidner *et al.* [21]. The same group implemented an hybrid reasoning mechanism adding task parameterization to their whole-body control and integrating symbolic transitions to concrete cleaning actions performed using a sponge [22,24]. Classical control approaches are the perfect solution in case the cleaning environment is controlled, well known a priori and does not change in time. However, if the robot faces unknown environments, it needs to adapt to unseen situations and to learn new skills. Classical controllers able to generalize to such unexpected situations are difficult to design.

In order to adapt to unknown environment and acquire new skills, cleaning robots should be able to learn from past and new experience. Using Reinforcement Learning (RL) robots can autonomously learn an approximation of optimal action policies for cleaning through self exploration of their action space. Hess *et al.* define an efficient state transition model for wiping table using a Markov Decision Process (MDP) [14]. The transition function is modeled by observing the outcomes of robot's actions and then used to generate paths for cleaning table surfaces. MDP is also used to clear objects from a table in fully-observable problems with uncertainty [27]. The same authors employ REX-D algorithm that integrates active teacher demonstration for increasing learning speed in order to sweep lentils from a plane [28]. Interactive RL approach with contextual affordances is developed by Cruz *et al.* to clean a table using state-action-reward-state-action (SARSA) [7]. In some cases the cleaning robot needs to handle high dimensional sensory data, like raw pixels data from camera images. In such situations deep reinforcement learning (Deep RL) models can simultaneously learn a desired behaviour from self exploration and extract the relevant features from raw data. Devin *et al.* [8] developed a Deep RL object-level attentional mechanism used to control a robot in different tasks like pouring almonds in a cup or sweeping citrus from a table. Moreover, Liu *et al.* proposed an imitation-from-observation algorithm used to perform various pushing and swiping actions. The model was trained both in simulation and on a real robot showing video recordings of the action from different viewpoints [26]. RL is a powerful tool that permits to find original solution to various control problems. Anyhow its flexibility comes with some drawbacks: long training time; exploration of dangerous states and configurations (*e.g.*,hitting a wall during navigation or colliding with the environment during

manipulation). Training in simulation can relax these problems, but performing a domain translation from simulation to the real world can be really complex.

To speed up the learning process and avoid dangerous situations, humans tend to exploit past experiences from other people (which performed similar actions) and to imitate their movements. In Learning from Demonstration (LfD) algorithms robot skills are derived from observations of human demonstrations and generalized to new environments. Dynamic Movement Primitives (DMPs) and Gaussian Mixture Models (GMM) are typically used to formalize and encode unit of action as a stable dynamical system with LfD.
Regarding the literature on DMPs, Ghalamzan *et al.* proposed an approach where DMP model and Inverse Optimal Control (IOC) are incorporated with a reward function to generate the necessary path in a new situation [35]. Kormushe *et al.* used DMPs and upper body kinesthetic demonstrations to teach to a robot how to clean a whiteboard [19]. In addition, a periodic DMP is applied to online coaching of robots in a human-robot interaction system [11]. Christopher *et al.* [6] show how weights of a periodic DMP can be learned using incremental locally weighted regression (ILWR). The periodic DMP is also used with force feedback for wiping differently tilted surfaces [10,11]. Moreover, in Pervez *et al.* [37] task parameterized DMP (TP-DMP) is used for adaptive motion encoding to perform a sweeping task based on few demonstrations.
On the GMM side, Calinon *et al.* [3] proposed the Task-Parameterized Gaussian mixture model (TP-GMM), a technique to generalize trajectories from demonstrated ones using task parameters (frames). Silvério *et al.* [40] combined TP-GMM and quaternion-based dynamical systems to learn full end-effector poses of a bimanual robotic manipulator to perform a sweeping task. A similar approach using partially observable task parameters without a dynamical system is proposed by Alizadeh *et al.* [1]. In their work, Hoyos *et al.* [16] extend TP-GMM with incremental learning skill. While several TP-GMM systems have been successfully used to generate robotic cleaning motions, none of them is able to autonomously learn the task parameters from raw sensory data (*i.e.* camera images).
One powerful solution to extract information from raw pixel data and learn important features on the images are Convolutional neural networks (CNNs). Rahmatizadeh *et al.* proposed a system able to learn multiple tasks using CNN and Long short-term memory (LSTM) networks [39]. The CNN plays the role of task selector and LSTM generates the robot joint command to send to the robot for cleaning small objects using a towel. Pervez *et al.* [38] proposed to use a CNN to learn the parameters of a TP-DMP directly from camera images, calling the system Deep-DMP (D-DMP). D-DMP was used to swipe different objects from a table.

In a recent work of our [18], we used a similar approach to learn the parameters of a TP-GMM to control a robot performing sweeping and wiping movements while cleaning a table. Two CNN based on AlexNet [20] are used to learn the parameters from raw input images, collecting the data through kinesthetic teaching. The main contribution compared to [38] is the ability of the system to generate different kinds of cleaning trajectories for different kinds of dirt: sweeping cluster of lentils and wiping off marker scribbles. A common limitation of both [38] and [18] is the need to retrain the system for different camera positions, environment to clean and robot to use. To solve these issues, in our last published work [4] we project the robot camera images into a canonical bird-view camera plane and we augment the dataset changing the illumination, shifting the images and applying Perlin noise [36] to the background. One CNN is used to directly predict means and covariances of a GMM, using GMR to obtain an estimation of the desired cleaning trajectory. After being trained with right arm's kinesthetic demonstrations, the robot was able to transfer his knowledge sweeping and wiping different kind of dirt using the left arm.

In this paper, we extend the works presented in [18] and [4] using a CNN/TP-GMM system, trained on a dataset collected on the iCub robot in Lisbon-Portugal, to control the Domestic Robot (DoRo) in Peccioli-Italy while cleaning a table. This type of transfer learning of a given task across different domains is known as transductive transfer learning [34] and also referred to as multi-robot transfer learning in the robotics community [13].

1.1 Contributions

To successfully transfer the knowledge gathered while training the iCub robot to the controlling task on the DoRo robot it is essential to collect a set of features that are invariant across domains, which is done using the techniques described in this paper, *e.g.*, viewpoint invariance using a virtual camera approach and dataset augmentation using Perlin noise, image translation and change in illumination. In addition, we perform a systematic analysis of the number of kinesthetic demonstration needed to successfully swipe and wipe off a table from lentils and marker scribbles, studying which type of data augmentation is more appropriate for our task. Thanks to this analysis we significantly reduced the high amount of kinesthetic demonstrations used in both [18] and [4].

In this paper, we adopted the same transformation to a canonical virtual camera used in [4]. We also used the kinesthetic demonstrations collected in [4] to create the new augmented dataset. Differently from [18] and [4], we calculated the hand orientations analytically. The orientations extracted from the kinesthetic demonstrations performed on the iCub are optimal for that particular robot and do not generalize well on the DoRo.

The main focus of this paper is to transfer the knowledge acquired by the iCub to the DoRo robot. For this reason, both the networks presented in [18] and [4] could have been used. The solution proposed in the former is more structured compared with the end-to-end solution of the later. Indeed, the combination of a CNN and a TP-GMM produces robust results and makes the network of [18] more suitable for this work.

The main contributions of this paper are:

1. **Demonstration of successful transferring of knowledge from a robot to another:** CNN and TP-GMM trained on the iCub are used to control the DoRo robot. To achieve this, three key generalization mechanisms are used:
   (a) Geometric image transformation (bird-eye view) to cope with different robot camera geometry (pose and calibration parameters). This was already proposed in [4].
   (b) Data augmentation to cope with different robot camera photometric properties (brightness, contrast, noise, color balance) and background clutter. This was already proposed in [4] but extended in this paper with a dual Perlin noise strategy.
   (c) End-efector orientation computed analytically to better adapt to different robot kinematics. This is a new method proposed in the current paper.
2. **Finding an optimal number of demonstrations needed to learn a cleaning motion:** CNN are trained with different number of kinesthetic demonstrations in order to detect a good compromise between size of the dataset and performance of the network.
3. **Proving the importance of domain randomization in our scenario:** Augmenting the dataset adding random Perlin noise to the background of the images is fundamental to generalize from iCub to DoRo.

1.2 Outline

The paper is organized as follows. Section II summarizes the proposed approach, describing the canonical virtual camera projection, showing the CNN architecture and giving a brief introduction to TP-GMM. Section III shows the experimental setup, while in Section IV the experimental results achieved are presented. Section V concludes the paper and gives some directions for further research.

2 Proposed approach

The goal of this paper is to transfer the knowledge acquired by the iCub robot in Lisbon, during a kinesthetic demonstration of a cleaning task, to the DoRo robot in Peccioli. Two different cleaning movements are taught to the iCub in order to clean a table: a sweeping motion to remove lentils from the table and a wiping motion to clean marker scribbles. The robot holds a sponge in its hand to perform the cleaning trajectories. In order to generalize to different robot camera positions and table heights, camera images are transformed to a canonical virtual image plane, similarly to what has been done in [4]. The canonical virtual camera is placed at a fix distance from the table, right on top of it, generating a bird-view image. Specific sizes and positions of objects placed on the table correspond to particular sizes and positions in the virtual image plane. From the virtual images, the robot estimates the correct cleaning hand trajectories using the same architecture introduced in our ICARSC 2018 paper [18]: a CNN estimate the initial, intermediate and final positions of the desired trajectory ($\mathbf{T}(n)$, $n = 0, \ldots, 200$) used to create the parameters of the TP-GMM (reference frames $\boldsymbol{X}_j = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$, $1 \leq j \leq 3$); GMR algorithm is used to estimate the desired trajectory $\mathbf{T}(n)$ from a TP-GMM defined by the reference frames $\boldsymbol{X}_j$. The only difference from [18] is the absence of the CNN to calculate the orientations of the initial, intermediate and final reference frames. High variation in the orientations of human demonstrations make impossible for the CNN to precisely predict the initial, intermediate and final orientations. To overcome this problem, in this paper we decided to analytically calculate the reference frames orientations from the reference frame positions predicted by a single CNN. Figure 2 shows the complete system architecture.

2.1 Virtual camera

The naive approach to use directly the unprocessed images taken from the robot cameras as input of the CNN has one important drawback. In real scenarios a robot approaches the table to clean it from different positions and with different head configurations. The pose of the

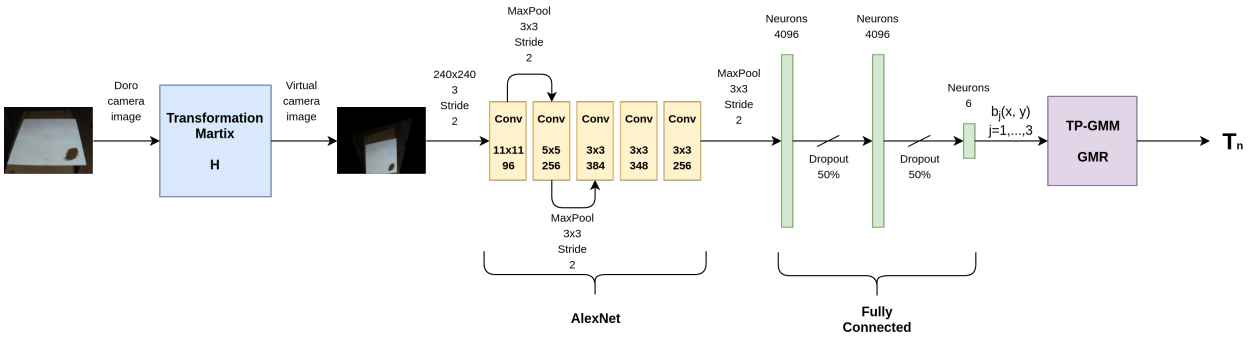


Fig. 2: System Architecture. Images from the robot's camera are transformed to virtual bird-view images. The virtual images are passed to a CNN that predicts initial, intermediate and final positions (b_1, b_2 and b_3). From the resulting TP-GMM the expected hand's trajectory is computed using GMR algorithm.

camera plane relative to the table plane during cleaning changes dynamically. This means that the CNN should intrinsically learn this spatial correlation directly from images. The task is not easy, and several demonstrations with different camera angles covering most of the possible configurations should be recorded. In our case this implies tens of thousands kinesthetic demonstration, something impossible to generate. One solution can be to fix a specific camera/table pose during training and place the robot always in the same configuration and position relative to the table during test. This is not a realistic scenario and, even if reasonable as proof of concept, such a system is not usable in real life.

In this paper we decided to use the approach adopted in [4]: apply a homographic transformation $\mathbf{H}$ to the robot camera plane in order to project it to a canonical virtual camera plane facing downward and placed right above the table at a fixed position.

This post-processing guarantees input images to be always taken from the viewpoint of the same canonical virtual camera, releasing the CNN to learn the geometric transformation between image plane and table plane. In order to generate the virtual camera image, the robot calculates an homographic transformation $\mathbf{H}$ from the robot camera plane to the virtual camera plane each time a new image is received:

$$z \begin{bmatrix} x_v \\ y_v \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} \tag{1}$$

where $\mathbf{P}_r = (x_r, y_r)$ and $\mathbf{P}_v = (x_v, y_v)$ are pixel coordinates in the real camera frame and virtual camera frame respectively and z is an arbitrary, non-zero scale factor.

Homography matrix $\mathbf{H}$ is calculated using the projection on both robot camera plane and virtual camera plane of 4 point laying on the table $\mathbf{P}(i) = (x(i), y(i), h_t)$, $i = 1, \ldots, 4$, where h_t is the table height in the robot reference frame. To obtain the 4 points $\mathbf{P}(i)$ the robot places his hand in 4 distinct positions on the table and uses his kinematics to extract the point planar coordinates $(x(i), y(i))$ and the high of the table h_t. The projection of $\mathbf{P}(i)$ on the robot camera plane $\mathbf{P}_r(i)$ is obtained using the body-eye forward kinematics and the intrinsic parameters of the camera:

$$z \begin{bmatrix} x_r(i) \\ y_r(i) \\ 1 \end{bmatrix} = K[I|\mathbf{0}] \cdot \tau_O^{eye}(q) \cdot \begin{bmatrix} x(i) \\ y(i) \\ h_t \\ 1 \end{bmatrix} \tag{2}$$

where $\tau_O^{eye}(q)$ denotes the Denavit-Hartenberg matrix from the robot reference frame O to the camera reference frame, I is a 3x3 identity matrix and $\mathbf{0}$ is a 3x1 vector of 0s.

To calculate the projections on the virtual camera plane $\mathbf{P}_v(i)$ we use the following function relating the robot reference frame O and the virtual camera image frame:

$$\mathbf{P}_v(i) = ((y(i) + 2/3)h, \quad (x(i) + 1)h) \tag{3}$$

where h is a scaling factor from pixel to meters that correspond to the height of the virtual camera image expressed in pixels. All points in 3D space are expressed on the iCub reference frame O placed near the hips of the robot. Ideally, the calibration process (*i.e.*,the robot touching 4 different positions on the table and extracting the points $\mathbf{P}(i)$) must be repeated every time the robot approaches a new table.

The procedure described above was used to generate the images of our dataset. In the case of DoRo, the head stays still during the entire cleaning experiment. For this reason we decided to skip equation 2 and select

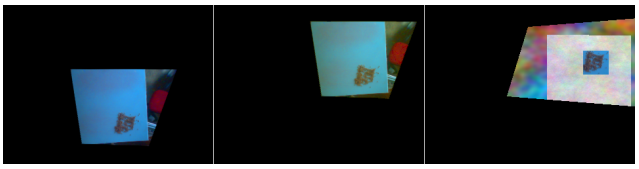


Fig. 3: Examples of data augmentation. Left: original dataset image. Center: same image with change in illumination and translation. Right: same image with Perlin noise table and background.

directly, from the camera image, the pixels corresponding to the points touched by the robot during calibration.

2.2 Data Augmentation

A second step necessary to achieve the transfer of cleaning capabilities from the iCub to the DoRo, was to perform a data augmentation on the original dataset of 659 elements. Specifically, three kind of data augmentation were performed (see Fig. 3 for an example):

1. **Changes in illumination:** a random value in between -0.15 and 0.15 is added to each RGB channel.
2. **Pixels and trajectories translation:** all pixels are translated in x and y with a random value ensuring that the dirt stays always visible inside the image. A correspondent shift in meters is then applied to the 200 trajectory points.
3. **Substituting background and table with Perlin noise:** we cropped the dirt from the image and substitute the background and the table with Perlin noise. Both projected field of view and table shape were slightly randomized. This data augmentation strategy is important to perform the transfer learning between the two robots, since the background is different.

To create the final dataset we augmented 10 times the original one applying changes in illumination and translations, plus more 10 times applying Perlin noise.

The original 659 elements are the same used to create the dataset in [4]. The difference is in the augmentation with the Perlin noise. Instead of substituting the entire background with a Perlin noise texture as in [4], we added two different Perlin noise textures, one for the table and one for the background (see Figure 3). In this way we are able to keep a basic structure of the environment inside the randomization.

2.3 End-effector Control

In our previous work [18], we used two CNNs to obtain three reference frames $\boldsymbol{X}_j, 1 \leq j \leq 3$ used as task parameters for the TP-GMM (one network for orientations $\boldsymbol{A}_j$ and the other one for positions $\boldsymbol{b}_j$ of the end-effector). Due to the high variability in kinesthetic demonstrations the orientation estimation error was high when evaluated on a test set comprising demonstrations that also presented a large variability in the orientation of the reference frames. As a consequence, in this work we decided to simplify the architecture and to use only one CNN to estimate the initial $\boldsymbol{b}_1$, intermediate $\boldsymbol{b}_2$ and final $\boldsymbol{b}_3$ positions, obtaining the rotation matrices $\hat{\boldsymbol{A}}_j$ directly from the estimated positions $\hat{\boldsymbol{b}_j}$. More precisely, $\hat{\boldsymbol{A}}_j$ are 2D rotation matrices of angles θ_j calculated as follow:

$$\begin{aligned} sin(\theta_j) &= \frac{y_{dist,j}}{\sqrt{{x_{dist,j}}^2+{y_{dist,j}}^2}} \\ cos(\theta_j) &= \frac{x_{dist,j}}{\sqrt{{x_{dist,j}}^2+{y_{dist,j}}^2}} \end{aligned} \qquad 1 \leq j \leq 3 \tag{4}$$

where $x_{dist,j}$ and $y_{dist,j}$ are the differences in x and y axis between initial and intermediate points in case of $j = 1$, and between intermediate and final points in case of $j = 2, 3$. Calculating the end-effector orientation this way is more flexible than extracting it from the kinesthetic data. The motion constraints of iCub and DoRo are quite different, same for the orientations achievable by each robot in a specific position. The orientations recorded moving the iCub's hand are biased by the kinematics costraints of the robot. Using equation 4 to calculate the 2D rotation matrices better generalize on a different robot with different constraints.

2.4 Convolutional neural network

The network architecture used in this paper is the same as in [18]. The CNNs architecture was devised based on the AlexNet [20] model changing only the output layer. In the proposed networks, the 1000 nodes output layer of the AlexNet is replaced with a fully connected one with 6 nodes. The outputs of the network are the x and y Cartesian coordinates of the three reference frames. The network takes as input a 3 channel (RGB) image resized to a dimension of 240x240 pixels. The network has a total of 8 layers: 5 convolutional layers and 3 feedforward fully connected layers. Fig. 2 depicts a detailed description of the network architecture. In order to train the network we generated a dataset of virtual camera images and trajectories using 659 kinesthetic demonstrations of wiping and sweeping movements. During

training we minimize the mean square error (MSE) between the network outputs and the initial, intermediate and final positions (x and y) of the hand trajectories. See Section 3.3 for a detailed description of this dataset.

2.5 Task parameterized Gaussian mixture model and Gaussian mixture regression

The use of a Gaussian Mixture Model to represent a set of trajectories performed by a human demonstrator is an efficient way of representing such demonstrations in a compact way, as all data points will be represented by a mixture of Gaussians that encompasses the average demonstrated trajectory, together with a corresponding variability. Under the LfD paradigm each demonstrated trajectory m from a set of M demonstrations consists of a set T_m vectors of dimension $D+1$, each vector $\boldsymbol{\xi}_n$ containing the observed task space variables y_n and the corresponding time t_n, for $1 \leq n \leq N$ and $N = \sum_{m=1}^{M} T_m$. By training a GMM with K components on this data set and then conditioning the resulting GMM on the time variable t_n one can obtain an average trajectory as a function of time, to be performed by the robot: this is known as Gaussian Mixture Regression (GMR) [12].

Task Parametrized GMM is an extension of GMM that allows the extrapolation of skills to different regions of the task space or to make such learned skill depend on a set of external variables, *e.g.*, a set of via points for the trajectory of the end-effector that the robot must reach in succession. This is done in [3] by considering a set of auxiliary frames of reference that define initial, intermediate and final points for the trajectory to perform. Each frame of reference j, $1 \leq j \leq P$, is represented by its origin $\boldsymbol{b}_{n,j}$ and rotation matrix $\boldsymbol{A}_{n,j}$.

We use the same framework as in [3] to learn to perform a cleaning movement from human demonstrations. Differently from that work we consider fixed frames of reference for each demonstration, as these are automatically calculated for each demonstration, and so we make its parameters depend solely on demonstration index m, *i.e.*, we use origin $\boldsymbol{b}_{m,j}$ and rotation matrix $\boldsymbol{A}_{m,j}$ instead. The Expectation-Maximization (EM) algorithm [29] is used to train the TP-GMM: the likelihood function to maximize is $p(\overline{\boldsymbol{\xi}}|\cdot) = \prod_{n=1}^{N} p(\boldsymbol{\xi}_n|\cdot)$, with

$$p(\boldsymbol{\xi}_n|\cdot) = \sum_{i=1}^{K} \pi_i p(\boldsymbol{\xi}_n|i) \ , \tag{5}$$

where π_i are the mixture proportions, $\overline{\boldsymbol{\xi}} = \{\boldsymbol{\xi}_n\}$ and $p(\boldsymbol{\xi}_n|i)$, the probability of mixture component i generating data point $\boldsymbol{\xi}_n$, is given by the joint distribution w.r.t. reference frames,

$$p(\boldsymbol{\xi}_n|i) = \prod_{j=1}^{P} p(\boldsymbol{\xi}_n|j,i) \ . \tag{6}$$

With $\boldsymbol{\xi}_n|j,i \sim \mathcal{N}\left(\boldsymbol{A}_{m,j}\boldsymbol{Z}_{i,j}^{\mu} + \boldsymbol{b}_{m,j}, \boldsymbol{A}_{m,j}\boldsymbol{Z}_{i,j}^{\Sigma}\boldsymbol{A}_{m,j}^{T}\right)$, this follows a normal distribution $\boldsymbol{\xi}_n|i \sim \mathcal{N}(\boldsymbol{\mu}_{m,i}, \boldsymbol{\Sigma}_{m,i})$, with

$$\Sigma_{m,i} = \left(\sum_{j=1}^{P} (\boldsymbol{A}_{m,j}\boldsymbol{Z}_{i,j}^{\Sigma}\boldsymbol{A}_{m,j}^{T})^{-1}\right)^{-1} \quad \text{and} \tag{7}$$

$$\mu_{m,i} = \Sigma_{m,i} \sum_{j=1}^{P} (\boldsymbol{A}_{m,j}\boldsymbol{Z}_{i,j}^{\Sigma}\boldsymbol{A}_{m,j}^{T})^{-1}(\boldsymbol{A}_{m,j}\boldsymbol{Z}_{i,j}^{\mu} + \boldsymbol{b}_{m,j}) \ , \tag{8}$$

where index m is the demonstration corresponding to data point $\boldsymbol{\xi}_n$.

Parameters $\boldsymbol{Z}_{i,j}^{\mu}$ and $\boldsymbol{Z}_{i,j}^{\Sigma}$ correspond to the mean vectors and covariance matrices describing a GMM for the data as seen from each frame of reference; together with π_i they correspond to the parameters to be learned using the EM algorithm. The most relevant feature of this approach is that in this process different weights are assigned to different frames of reference, according to the current time of the reproduction, thus effectively capturing the most relevant features of the human demonstrations. These correspond to some invariance of the demonstrations as seen from each frame of reference, encoded in a low variance estimate for the task space variables, taken from the corresponding GMM.

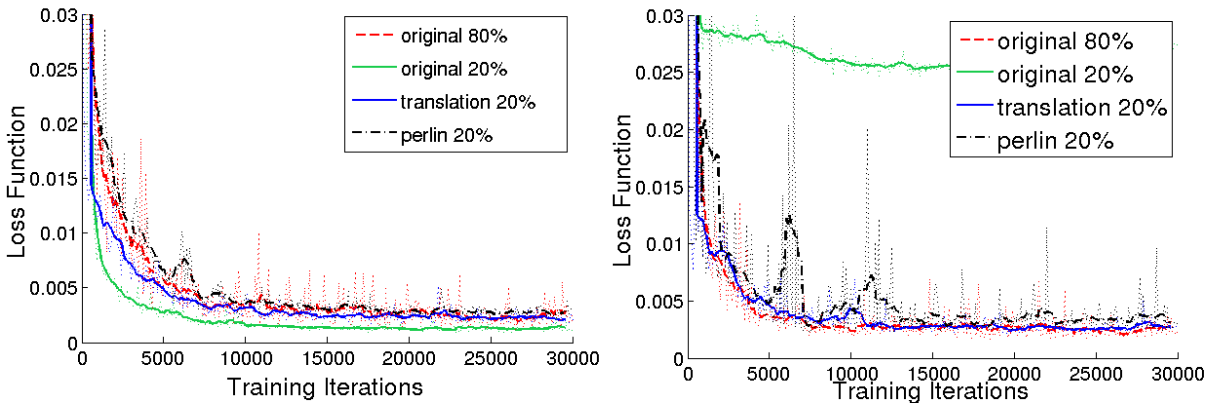
The EM training procedure finds, in the E-Step, responsibilities

$$\gamma_{n,i} = \frac{\pi_i p(\boldsymbol{\xi}_n|i)}{\sum_{k=1}^{K} \pi_k p(\boldsymbol{\xi}_n|k)} \tag{9}$$

and uses these values to update estimates for parameters $\boldsymbol{Z}_{i,j}^{\mu}$, $\boldsymbol{Z}_{i,j}^{\Sigma}$ and π_i in the M-Step (for more details please refer to [3]). After learning, given a new set of frames of reference $\boldsymbol{X}_j = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$, provided by the neural network from the test image, a trajectory $\boldsymbol{T}_n$ is generated in the task space by conditioning the distribution $p(\boldsymbol{\xi}|\cdot)$ on the time variable t_n, using (5), (7) and (8).

3 Experimental Setup

In this section, we will describe the two robots used on this work, the dataset collected and how the system was initialized. The iCub robot was used to collect



(a) Training Loss (b) Validation Loss

Fig. 4: Training and Validation Loss of the 4 used Network. (Best viewed in color)

the dataset using zero torque controllers to perform the kinesthetic teaching demonstrations, while the DoRo robot was used to test the system in a real world scenario accessing the transferring capabilities of the proposed architecture.

3.1 iCub Description

The iCub humanoid robot [30] has 53 motors that move the hands, arms, head, waist and legs. Regarding the sensory capabilities, it has a stereo vision system (cameras in the eyeballs), touch (tactile fingertips and artificial skin), vestibular sensing (IMU on top of the head) and proprioception (motor encoders and torque sensors), which are major features that allowed us to record the dataset used in this article.

3.2 DoRo Description

The domestic robot (DoRo) [5] is a service robot moved by a SCITOS G5 mobile platform (developed by Metralabs). A Kinova Jaco arm (6 DoF manipulator integrated 3-DOF hand) is mounted on the right side to perform manipulation tasks. On board are present a front laser (SICK S300) and a rear laser (Hokuyo URG-04LX) to view and avoid obstacles and perform self-localization. A pan-tilt system is installed on the head with two high-res cameras equipped with different lenses, and an Asus Xtion Pro RGB-Depth camera used for object detection. The eyes are equipped with multicolor LEDs and a speaker is used to interact with the users.

3.3 Collecting the Dataset

In this work we used a dataset composed by 659 kinesthetic demonstrations collected in [4] changing the data augmentation strategies (see Section 2.2 for more details). To collect the dataset we placed the iCub robot, holding a sponge on its right hand, in front of a white table of size 50x50 cm. For each demonstration some dirt was placed on the table (lentils clusters or marker scribbles). A human guided the iCub right hand cleaning as much as possible of the dirt spot with a specific motion for each dirt type. The inputs of the dataset are images of the dirty table recorded from the iCub right eye before to perform each kinesthetic demonstration and the labels are the right hand 2D trajectories in x and y of the robot reference frame. Each trajectory consists of 200 elements. This dataset was then augmented using the procedure explained in Section 2.2 resulting in a new dataset 20 times bigger. In order to train the CNN we extracted from the trajectories the first, intermediate and final points.

3.4 System initialization

Fig.5 depicts the workspace used in the experiments with DoRo. The system setup was initialized with the DoRo robot with its head and arm in a pre-defined position. Indeed, this initial position can be changed without losing generality and without the need of re-training the whole system. Moreover, a 50x50 cm table was placed in front of the robot at different heights: 67 cm, 70 cm and 79 cm. The z-coordinate used when generating the final end-effector trajectory was pre-defined matching the measured table height. Although, the table height was measured by the experimenter in the

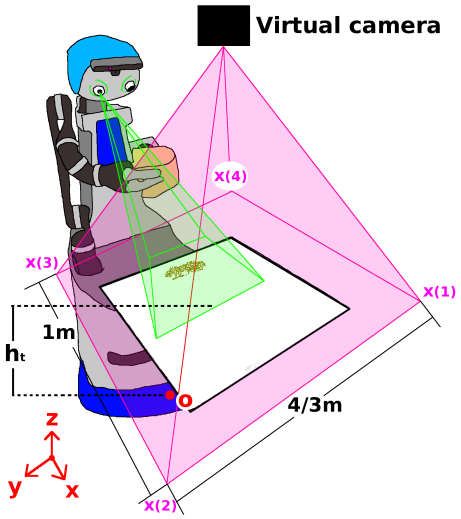


Fig. 5: The workspace used in our experiments. The Doro robot is placed in front of a table. h_t is the height of the table expressed in the robot reference frame O. A bird-view virtual camera is placed on top of the table.

DoRo case, it is possible to use a calibration routine exploiting, for instance, touch or torque sensors on the hand to feel the table and extract this information. Furthermore, the virtual camera was calibrated placing the robot arm on the table in four different positions as described in the Virtual Camera section (see Section 2.1) using a joy-stick controller. To obtain similar images to the one collected on the iCub, we used the same table relative position to the virtual camera adopted in [4]. As long as the proportion between meters and virtual camera pixels is the same as in the iCub generated dataset, the size of the table is not relevant. Due to the DoRo kinematic structure, we adopted a top grasp configuration to hold the sponge. The use of task space coordinates during the learning phase provides invariance to a particular robot kinematics model, as long as the 3 reference frames provided by the CNN are sufficiently far from the robot kinematics singularities. In our experiments, this is achieved by constraining the region of the table to be cleaned to be in the robot reachable space.

The collected dataset was divided in two sets: i) training and ii) validation. The validation set was defined as 20% of the original dataset (*i.e.*, 20% of 659 human demonstrations) and the training set was build selecting the remaining 80% and performing data augmentation. The six position labels (*i.e.*, the reference frames for the TP-GMM - $\mathbf{X}_j$) were normalized to improve the learning and the mean image of the training set was also calculated and subtracted from the input image on each training example. The network was implemented using Caffe [17] and trained with the Adam optimizer with a fixed learning rate of 0.001 and dropout with ratio 0.5 on the first two fully-connected layers (see Fig. 2). Moreover, the training process used a batch size of 80 and was stopped after 30000 iterations (about 1000 epochs).

3.5 Evaluation method

The system was tested placing a dirt spot (marker or lentils) on the table and making the robot clean it in 5 repetitions without human intervention. The evaluation of the cleaning task is defined according to the different type of dirt presented on the environment. For the case of marker scribbles, the percentage of dirty area $m_1(r)$ after each repetition was calculated:

$$m_1(r) = \frac{A(r)}{A(1)} 100, \qquad r = 1, \ldots, N_r, \tag{10}$$

where $A(r)$ is defined as the dirty area in pixel at repetition r, and $N_r = 5$ is the number of repetitions.

For the lentils case, the performance is evaluated using the metric $m_2(r)$, which is defined by the following expression:

$$m_2(r) = \frac{D(r)}{D(1)} 100, \qquad r = 1, \ldots, N_r, \tag{11}$$

where $D(r)$ measures how far the dirt is from the target position. $D(r)$ is defined as:

$$D(r) = \sum_{i=1}^{N_p} I(i) \times \sqrt{(i-o)^T (i-o)}, \tag{12}$$

where $I(i)$ is an indicator function which identifies the dirty pixels, o is the bottom-right corner of the table (the target position) expressed in pixels and N_p is the total number of pixels in the input image.

To calculate the dirty area in the images, we used a post-processing phase where a color (RGB-based) segmentation was performed.

4 Results

In this section we present the results of the proposed cleaning architecture showing: i) a detailed analysis of the performance of the neural network and of the TP-GMM correlating it with the amount of training examples used and data augmentation strategies and ii) real-world experiments testing the learned architecture on a different robotic platform - the DoRo robot (see Fig.1).

	80% (527 original samples)			**50%** (330 original samples)			**20%** (132 original samples)			**10%** (66 original samples)		
	O	**T**	**P**	**O**	**T**	**P**	**O**	**T**	**P**	**O**	**T**	**P**
Effective Training Samples	527	5797	11067	330	3630	6930	132	1452	2772	66	726	1386
Training Loss [$^{\times 10^{-3}}$]	**2.70**	2.96	3.04	1.25	2.43	2.76	**1.29**	**2.09**	**2.81**	1.18	2.17	3.66
Validation Loss [$^{\times 10^{-3}}$]	**2.65**	1.65	1.94	14.5	2.01	2.64	**27.54**	**2.76**	**3.15**	45.21	5.02	4.10

Table 1: Loss after training the Network for 30000 iterations. The dataset is composed with 80%, 50%, 20% and 10% of the initial dataset (527, 330, 132 and 66 samples, respectively) to train the Network using three types of data. (**O**: Original examples; **T**: translation and illumination changes and **O** examples; **P**: Perlin noise augmentation and **T** examples, check Section 2.2)

4.1 Validation set

The results presented on this section were evaluated on (the same) 20% of the original iCub cleaning dataset, which we call for now on as *Validation Set*. We exploit the 80% remaining examples to train the network and the TP-GMM algorithm, testing several strategies of: i) data augmentation (for the case of the network) and ii) different quantities of training data to access the amount of necessary examples to achieve a good accuracy in the cleaning task (on both, CNN and TP-GMM).

4.1.1 Network tests

The execution of kinesthetic demonstrations to feed the system with learning examples could be time consuming, so to access the performance of the Network on the validation set according to the data present in the training set, we run the Network several times with different types of data augmentation and with different amounts of initial kinesthetic teaching examples.

We have created 12 (different) training sets combining four (4) percentages of the original dataset (80% - 527 samples, 50% - 330 samples, 20% - 132 samples and 10% - 66 samples) with three (3) data types (**O**, **T** and **P**). The datasets of type **O** include only the Original samples, the datasets **T** extends **O** adding 2 data augmentation strategies (variations of illumination and Translation) and the **P** datasets include the previous ones adding the Perlin noise images as well. The performance of the Network on the validation set taking into consideration the amount of data used and augmentation strategy performed can be seen in Table 1.

The training loss is similar in all the training sets which implies that the Network is learning (*i.e.*, reducing the error) on those datasets, however, the validation loss increases when we feed the network with less examples. For instance, using only 20% of the original dataset (**O20%**), the loss increases one order of magnitude (from 2.65 to 27.54, on **O80%** and **O20%**, respectively). Looking on the data augmentation strategies (**T** and **P**) using only 20% of the available data, one can see that the validation loss is similar to the case the network is trained on 80% of the original dataset is used (**O80%** = 2.65; **T20%**=2.76 and **P20%**=3.15). Comparing the most promising networks with data augmentation (**T20%** and **P20%**) with the networks trained on the original dataset (**O80%** and **O20%**), one can see the evolution of the loss on training and validation on Fig. 4. In Fig. 4, the solid line is the filtered loss signal using a moving average filter with a window size of 5 (*i.e.*, 500 iterations) and the dotted signal is the original (non-filtered) data. Furthermore, with only 10% of the kinesthetic teaching examples (66 samples), the network is not able to generalize well, achieving a validation error 2 times bigger.

Clearly, the Perlin noise is not essential on the validation set (achieving a similar validation loss). This happens because the background is roughly the same (the environment did not change). However, as can be seen in Section 4.2, it will be essential when generalizing to another background (on the DoRo robot). After this evaluation, we conclude that **T20%** and **P20%** are suitable to test on the real robot and are a good trade-off between number of kinesthetic teaching and accuracy achieved.

4.1.2 TP-GMM tests

The TP-GMM should be learned using cleaning trajectories as demonstrations. In order to access the amount of demonstrations needed to learn to generate the task trajectories we use the 80% of the original dataset as training and 20% of the dataset as validation set (the same validation set in Sec. 4.1.1). The TP-GMM was initialized using several quantities of random sampled demonstrations and the learned model was tested on the validation set. Fig. 6 depicts the mean and standard deviation on 10 trials (increase each 10 demonstrations) of the error between GMR generated trajectories and kinesthetic ones as they vary in the number of demonstrations used.

	Marker		Lentils	
	Area cleaned	Standard Deviation	Distance reduced	Standard Deviation
Cauli et al. (iCub) [4]	80%	15%	45%	2%
Our results (DoRo)	75%	20%	50%	10%

Table 2: Comparison between the previous results of [4] on the iCub robot and our results on the DoRo robot. The test scenario is the same for both the experiments. The two systems have a different network architecture and a different data augmentation strategy.

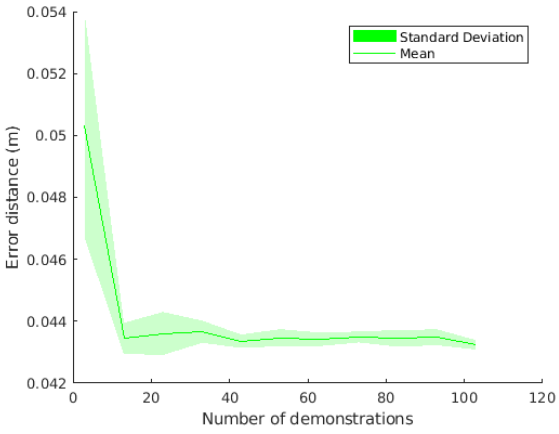


Fig. 6: Variation of the error between GMR generated trajectories and kinesthetic ones, with the number of demonstrations presented to the GMM.

4.2 Robot experiments

The proposed architecture was tested on a real scenario using the DoRo robot (See Fig. 1) to determine the transferring capabilities of the cleaning system to a different robotic platform. The robot should try to clean the dirty table (with marker scribbles or cluster of lentils) using a budget of five (5) repetitions ($N_r = 5$). For each repetition (r), the agent looks to the table, detects the dirt and adjusts the output trajectory accordingly (same experimental setup as in [4] for the iCub robot). In Fig. 7, one example of cleaning markers scribbles (left column) and one of cluster of lentils (right column) can be seen with the output trajectory super-imposed on the image with black color using the P20% Network. In this case, the red ink was cleaned after the second repetition and the cluster of lentils is closer to the right bottom corner of the table after the five repetitions (final result). Note that, we did not draw the trajectories generated on the marker scribbles inside the five repetitions budget (*i.e.*, $r = 3, 4, 5$), since the table was already clean (apart from some small fragments invisible for the robot). Moreover, our architecture does not have a criteria to stop the cleaning task (see Section 5 for further discussion), so the robot will perform always the same trajectory if an input image with a clean table is shown.

In a more quantitative analysis, and using the error metric 1 defined on Eq. (10), the DoRo robot performed 15 cleaning experiments on marker scribbles setting the table at 3 different heights. The results over the 5 repetitions budget can be seen in Fig. 8. We reduced the dirt in 75% of its initial area with a standard deviation of 20%. In the lentils case, the table was set at the same 3 different heights and the robot performed 15 different experiments. The mean error and standard deviation using the metric m_2 (see Eq (11)) can be seen in Fig. 9. The percentage of the initial distance from the bottom right corner of the table (the target point when cleaning this type of dirt) was reduced in 50% with a standard deviation of 10%.

Table 2 shows the comparison between the results obtained on the DoRo and the results of [4] obtained on the iCub (networks and datasets of the 2 systems have some differences. Please refer to sections 1.1 and 3.3). The results on the DoRo are close to the results obtained on the iCub, showing how a system trained on one robot can be used to control a second one.

The networks trained on **O80%**, **T20%** and **P20%** (check Fig. 4 and Table 1) were tested on the DoRo robot. The network trained with only the original images (**O80%**) was not able to detect and clean any type of dirt. The **T20%** network was able to clean the dirt when its location was on the central part of the table but with lower overall performance. Indeed, the data augmentation with Perlin noise (**P20%**) was essential for transferring the learned cleaning movements from the iCub to the DoRo robot, since the background surrounding the robot is completely different on the DoRo robot (see examples in Fig. 7) and on the iCub (see original dataset on Fig. 3 - left).

5 Conclusions and Future work

We presented a framework for learning how to perform a given cleaning task from human kinesthetic demonstrations, directly from raw camera images, and later transferring the knowledge gathered in this process to a different robot. The parameters of the convolutional

(a) r = 1 (b) r = 1

(c) r = 2 (d) r = 2

(e) r = 3 (f) r = 3

(g) r = 4 (h) r = 4

(i) r = 5 (j) r = 5

(k) final result (l) final result

Fig. 7: Testing examples on the real robot - DoRo - over 5 budget repetitions. In black color it is possible to see the output trajectory. Left Column: marker scribbles; Right Column: lentils.

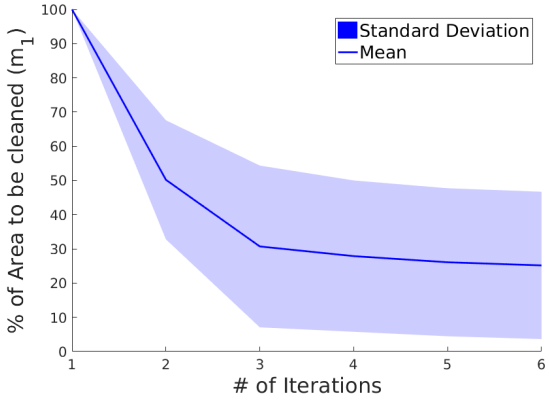


Fig. 8: Mean and standard deviation, using evaluation metric 1 (See Eq. (10)), for 15 Marker Experiments with 3 different table height on the DoRo Robot.

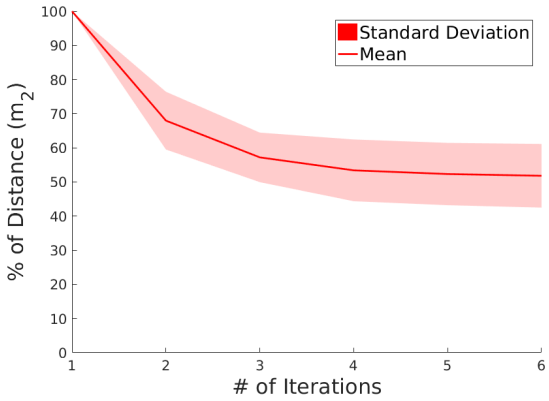


Fig. 9: Mean and standard deviation, using evaluation metric 2 (See Eq. (11)), for 15 Lentils Experiments with 3 different table height on the DoRo Robot.

neural network trained using the provided demonstrations can be directly used in a different robot if some care is taken to make the network invariant to illumination and perspective changes when applied to a different robot. To achieve this we employ several techniques, such as using a virtual camera to achieve perspective invariance across robots and data augmentation by random changes in illumination, image translation and adding Perlin noise to the background regions of the images. The use of these strategies reduced the need for a large training set — only 20% of the recorded data was needed to achieve a similar test error when compared to the situation where no data augmentation was used. Furthermore, the robustness provided by the use of these techniques allowed for a straightforward use of the trained CNN in a different robot: the trained CNN using data acquired from human demonstrations on the iCub robot was used in the DoRo platform to

perform the same task without noticeable loss of performance.

A current limitation of the proposed framework is the need to perform an initial manual calibration to set up the virtual camera on the second robot, so that the learned CNN can be used to perform the cleaning task. Also, currently the trajectory generation is performed in open-loop, with intermediate trajectory points provided by the network learned from human demonstrations. As a consequence, the robot can become stuck performing the same trajectory over and over again when the performed movement does not significantly change the dirt configuration: this typically happens when the table is almost cleaned, as discussed in the previous section Implementing a stopping criteria based on the detection of a clean table could be a straightforward solution for this problem. Moreover, a direction for further research that can alleviate this issue is to use a deep reinforcement learning approach [8], where the robot can learn from trial and error how to clean the table, based on a set of image features provided by a deep neural network.

Acknowledgements This work was partially supported by Fundação para a Ciência e a Tecnologia (project UID/EEA/-50009/2013 and Grant PD/BD/135115/2017) and the RBCog-Lab research infrastructure. We acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research.

References

1. Alizadeh, T., Calinon, S., Caldwell, D.G.: Learning from demonstrations with partially observable task parameters. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 3309–3314. IEEE (2014)
2. Cakmak, M., Takayama, L.: Towards a comprehensive chore list for domestic robots. In: Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction, pp. 93–94. IEEE Press (2013)
3. Calinon, S., Alizadeh, T., Caldwell, D.G.: On improving the extrapolation capability of task-parameterized movement models. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 610–616. IEEE (2013)
4. Cauli, N., Vicente, P., Kim, J., Damas, B., Bernardino, A., Cavallo, F., Santos-Victor, J.: Autonomous table-cleaning from kinesthetic demonstrations using deep learning. In: Joint IEEE International Conference on Development and Learning (ICDL) and Epigenetic Robotics (EpiRob). IEEE (2018)
5. Cavallo, F., Limosani, R., Manzi, A., Bonaccorsi, M., Esposito, R., Di Rocco, M., Pecora, F., Teti, G., Saffiotti, A., Dario, P.: Development of a socially believable multi-robot solution from town to home. Cognitive Computation **6**(4), 954–967 (2014)
6. Christopher, A., Andrew, M., Stefan, S.: Locally weighted learning. Artif Intell Rev **11**(1-5), 11–73 (1997)
7. Cruz, F., Magg, S., Weber, C., Wermter, S.: Training agents with interactive reinforcement learning and contextual affordances. IEEE Transactions on Cognitive and Developmental Systems **8**(4), 271–284 (2016)
8. Devin, C., Abbeel, P., Darrell, T., Levine, S.: Deep object-centric representations for generalizable robot learning. arXiv preprint arXiv:1708.04225 (2017)
9. Dornhege, C., Hertle, A.: Integrated symbolic planning in the tidyup-robot project. In: AAAI Spring Symposium: Designing Intelligent Robots (2013)
10. Gams, A., Petrič, T., Do, M., Nemec, B., Morimoto, J., Asfour, T., Ude, A.: Adaptation and coaching of periodic motion primitives through physical and visual interaction. Robotics and Autonomous Systems **75**, 340–351 (2016)
11. Gams, A., Ude, A.: On-line coaching of robots through visual and physical interaction: Analysis of effectiveness of human-robot interaction strategies. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 3028–3034. IEEE (2016)
12. Ghahraiuani, Z.: Solving inverse problems using an em approach to density estimation. In: Proceedings of the 1993 Connectionist Models summer school, p. 316. Psychology Press (1994)
13. Helwa, M.K., Schoellig, A.P.: Multi-robot transfer learning: A dynamical system perspective. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4702–4708. IEEE (2017)
14. Hess, J., Sturm, J., Burgard, W.: Learning the state transition model to efficiently clean surfaces with mobile manipulation robots. In: Proc. of the Workshop on Manipulation under Uncertainty at the IEEE Int. Conf. on Robotics and Automation (ICRA) (2011)
15. Hess, J., Tipaldi, G.D., Burgard, W.: Null space optimization for effective coverage of 3d surfaces using redundant manipulators. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 1923–1928. IEEE (2012)
16. Hoyos, J., Prieto, F., Alenya, G., Torras, C.: Incremental learning of skills in a task-parameterized gaussian mixture model. Journal of Intelligent & Robotic Systems **82**(1), 81–99 (2016)
17. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on Multimedia, pp. 675–678. ACM (2014)
18. Kim, J., Cauli, N., Vicente, P., Damas, B., Cavallo, F., Santos-Victor, J.: icub, clean the table! a robot learning from demonstration approach using deep neural networks. In: IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 3–9. IEEE (2018)
19. Kormushev, P., Nenchev, D.N., Calinon, S., Caldwell, D.G.: Upper-body kinesthetic teaching of a free-standing humanoid robot. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 3970–3975. IEEE (2011)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
21. Leidner, D., Beetz, M.: Inferring the effects of wiping motions based on haptic perception. In: Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on, pp. 461–468. IEEE (2016)

22. Leidner, D., Bejjani, W., Albu-Schäffer, A., Beetz, M.: Robotic agents representing, reasoning, and executing wiping tasks for daily household chores. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, pp. 1006–1014. International Foundation for Autonomous Agents and Multiagent Systems (2016)
23. Leidner, D., Borst, C., Dietrich, A., Beetz, M., Albu-Schäffer, A.: Classifying compliant manipulation tasks for automated planning in robotics. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pp. 1769–1776. IEEE (2015)
24. Leidner, D., Dietrich, A., Beetz, M., Albu-Schäffer, A.: Knowledge-enabled parameterization of whole-body control strategies for compliant service robots. Autonomous Robots **40**(3), 519–536 (2016)
25. Liang, J., Zhang, G., Wang, W., Hou, Z., Li, J., Wang, X., Han, C.S.: Dual quaternion based kinematic control for yumi dual arm robot. In: Ubiquitous Robots and Ambient Intelligence (URAI), 2017 14th International Conference on, pp. 114–118. IEEE (2017)
26. Liu, Y., Gupta, A., Abbeel, P., Levine, S.: Imitation from observation: Learning to imitate behaviors from raw video via context translation. arXiv preprint arXiv:1707.03374 (2017)
27. Martínez, D., Alenya, G., Torras, C.: Safe robot execution in model-based reinforcement learning. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pp. 6422–6427. IEEE (2015)
28. Martínez, D., Alenya, G., Torras, C.: Relational reinforcement learning with guided demonstrations. Artificial Intelligence **247**, 295–312 (2017)
29. McLachlan, G., Krishnan, T.: The EM algorithm and extensions. John Wiley & Sons (1997)
30. Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., Montesano, L.: The iCub humanoid robot: an open-systems platform for research in cognitive development. Neural Networks **23** (2010)
31. Nebel, B., Dornhege, C., Hertle, A.: How much does a household robot need to know in order to tidy up. In: Proceedings of the AAAI Workshop on Intelligent Robotic Systems, Bellevue, WA (2013)
32. Okada, K., Kojima, M., Sagawa, Y., Ichino, T., Sato, K., Inaba, M.: Vision based behavior verification system of humanoid robot for daily environment tasks. In: Humanoid Robots, 2006 6th IEEE-RAS International Conference on, pp. 7–12. IEEE (2006)
33. Ortenzi, V., Adjigble, M., Kuo, J.A., Stolkin, R., Mistry, M.: An experimental study of robot control during environmental contacts based on projected operational space dynamics. In: Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on, pp. 407–412. IEEE (2014)
34. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2010)
35. Paxton, C., Hager, G.D., Bascetta, L., et al.: An incremental approach to learning generalizable robot tasks from human demonstration. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 5616–5621. IEEE (2015)
36. Perlin, K.: An image synthesizer. ACM Siggraph Computer Graphics **19**(3), 287–296 (1985)
37. Pervez, A., Lee, D.: Learning task-parameterized dynamic movement primitives using mixture of gmms. Intelligent Service Robotics **11**(1), 61–78 (2018)
38. Pervez, A., Mao, Y., Lee, D.: Learning deep movement primitives using convolutional neural networks. In: IEEE-RAS International Conference on Humanoid Robots (2017)
39. Rahmatizadeh, R., Abolghasemi, P., Bölöni, L., Levine, S.: Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. arXiv preprint arXiv:1707.02920 (2017)
40. Silvério, J., Rozo, L., Calinon, S., Caldwell, D.G.: Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pp. 464–470. IEEE (2015)
41. Urbanek, H., Albu-Schaffer, A., van der Smagt, P.: Learning from demonstration: repetitive movements for autonomous service robotics. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 4, pp. 3495–3500. IEEE (2004)
42. Yamazaki, K., Ueda, R., Nozawa, S., Mori, Y., Maki, T., Hatao, N., Okada, K., Inaba, M.: System integration of a daily assistive robot and its application to tidying and cleaning rooms. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 1365–1371. IEEE (2010)