Autonomous table-cleaning from kinesthetic demonstrations using Deep Learning

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

Abstract— We address the problem of teaching a robot how to autonomously perform table-cleaning tasks in a robust way. In particular, we focus on wiping and sweeping a table with a tool (*e.g.*, a sponge). For the training phase, we use a set of kinestethic demonstrations performed over a table. The recorded 2D table-space trajectories, together with the images acquired by the robot, are used to train a deep convolutional network that automatically learns the parameters of a Gaussian Mixture Model that represents the hand movement. After the learning stage, the network is fed with the current image showing the location/shape of the dirt or stain to clean. The robot is able to perform cleaning arm-movements, obtained through Gaussian Mixture Regression using the mixture parameters provided by the network. Invariance to the robot posture is achieved by applying a plane-projective transformation before inputting the images to the neural network; robustness to illumination changes and other disturbances is increased by considering an augmented data set. This improves the generalization properties of the neural network, enabling for instance its use with the left arm after being trained using trajectories acquired with the right arm. The system was tested on the iCub robot generating a cleaning behaviour similar to the one of human demonstrators.

I. INTRODUCTION

Robots capable of working alongside with humans and performing simple and mildly complex tasks in a full autonomous way are increasingly becoming a key research topic in the robotics field [1]. Performing household chores, like preparing a meal, cleaning a room, or doing the laundry, is becoming more and more relevant in a robotic context. The increasing number of human resources required to adequately support a growing elderly population, could be progressively complemented by autonomous service robots capable of responding to such demands. While the social interaction abilities of such kind of robots are gradually developing into a mature stage [2], there is still a long way to go with respect to their physical interaction and manipulation abilities [3].

The main difficulty is still the lack of robustness of such robots when interacting in real world settings: while they can succeed in highly controlled environments, the presence of disturbances or uncertainty can easily degrade their performance. For instance, slight changes on objects shapes and positions can drastically lower the manipulation

[1]Institute for Systems and Robotics, Instituto Superior Tecnico, Universidade de Lisboa, Portugal {`ncauli,pvicente,bdamas,alex,jasv`} `@isr.tecnico.ulisboa.pt`

[2]BioRobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy. `j.kim@sssup.it`, `filippo.cavallo@santannapisa.it`

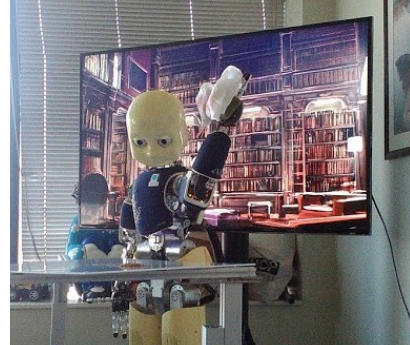[3]CINAV — Centro de Investigação Naval, Almada, Portugal

Fig. 1. The iCub humanoid robot using a tool to clean a table facing him.

performance of the robot interacting with those objects; changes in illumination and viewpoint can seriously hinder the visual recognition and tracking capabilities; and even when a robot is successful at performing some given task it is usually difficult to transfer such skill to a robot with different kinematic and sensor characteristics.

We propose an approach to improve the robustness with respect to the aforementioned issues, using a cleaning task to demonstrate the validity of this procedure. Cleaning tasks have received some attention by the robotics community: Okada *et al.* apply an inverse kinematics based programming approach to compute whole-body motions for the tasks of dish washing, sweeping and vacuuming the floor using a humanoid robot [4]; a simulation-based approach using a temporal projection system is applied to a table sponge wiping task [5]; In [6] an efficient approach based on null-space optimization is used to generate cleaning trajectories; and a path planning algorithm based on task decomposition is used for wiping tasks in [7], to name just a few examples.

A different approach, based on adaptive behavior, can in principle present some robustness to disturbances and uncertainty: Cruz *et al.*, for instance, use a reinforcement learning (RL) framework and contextual affordances to learn how to perform a cleaning task [8]. To overcome the typical slow convergence rate of RL schemes, the Learning from Demonstration (LfD) paradigm can be used to provide a good starting point to the RL iterations: it is critical in non-simulated scenarios to avoid the initial random exploration phase. Some works use LfD to perform cleaning tasks by kinesthetic teaching [9]: however, without further learning these pure imitation learning schemes are not robust to uncertainties, being able to mimic previously seen movements
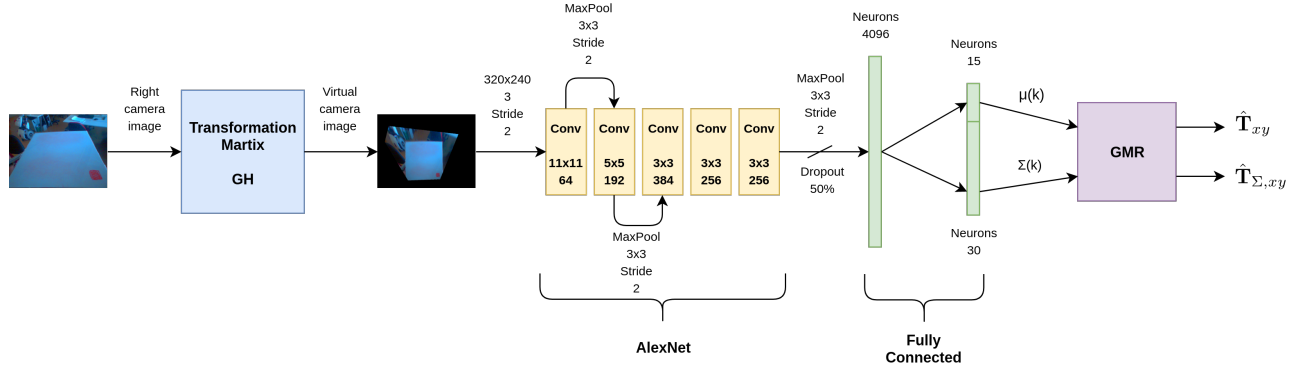
Fig. 2. System Architecture. Images from robot's right camera are transformed to virtual bird-view images. The virtual images are passed to a CNN that predicts $\boldsymbol{\mu}$ and Σ of 5 gaussians. From those the expected hand's trajectory is computed using the GMR algorithm.

only. To overcome this issue a task-parameterized Gaussian mixture model framework is proposed that is able to generalize to unseen movements, characterized by different reference frames defined by the locations of the objects that serve as markers [10], [11], using kinesthetic teaching to demonstrate how to perform a dust sweeping task.

In [12] the need to use markers of any kind to signal the task reference frames in [10], [11] is removed, using a convolutional neural network (CNN) architecture [13] that directly learns the start, intermediate and final frames that implicitly define the trajectory to perform from the raw images and human demonstrations. This allows the robot to perform the cleaning task autonomously, without the physical presence of such markers or any human intervention. This paper builds upon the previous work presented in [12] and tries to simplify the learning process while, at the same time, introduces some additional mechanisms that improve the robustness of the approach. In this paper a CNN is used to directly learn a Gaussian Mixture Model (GMM) that represents the task space trajectory to perform from the image data and human demonstrations. This eliminates an intermediate learning phase present in [12], where an Expectation-Maximization procedure was used to fit a mixture of Gaussians to the demonstrated task space trajectories; after the learning phase a task was performed by marginalizing the mixture model with respect to mixture components and frames of reference, as provided by the CNN from the raw images. In this work such reference frames are no longer needed, as the CNN directly provides a GMM describing the trajectory to be performed from the camera images that hopefully will succeed at the cleaning task.

In this work we apply some techniques that improve the applicability of our method to different scenarios and robots. First, a plane-projective transformation is applied to the acquired images before presenting them to the CNN. This transformation corresponds to creating a virtual camera pointed downward and placed above the robot working area, and helps achieving invariance in the acquired images with respect to perspective changes due to robot different postures relatively to the working area. Additionally, a data augmentation is performed using Perlin noise [14] and additive random noise: this improves the CNN generalization abilities, achieving a better robustness to illumination changes, and also helps reduce the number of collected demonstrations required for training the CNN. Overall, all these improvements enhance the robustness of the previous architecture to disturbances and we demonstrate this feature by using the left arm of iCub, the robot used in the experiments, to perform sweeping and wiping movements using the CNN learned using its right arm. This is a very simple case of transfer learning [15], but the use of such perspective transformation can in principle make possible the transfer of the learned model to a different robot. The acquired dataset with all the demonstrations will be publicly available[1].

II. PROPOSED APPROACH

To train the CNN used in this architecture, a dataset of 659 examples was created guiding the right hand of the robot while cleaning a table from marker scribbles and clusters of lentils. In order to perform the task a sponge was attached to the right hand of the robot. The data recorded during each demonstration was: right camera images showing the initial state of the table, encoders values of all robot joints during the entire movement, 2D trajectory of the right hand calculated by the iCub cartesian solver [16]. After training, to perform a cleaning trajectory by the robot arm that is independent from the robot posture and the height of the table, first the raw images are pre-processed as follows: encoders data q, camera's intrinsic parameters matrix K and table height h_t are used to calculate a transformation matrix $\mathbf{H}$ between the camera images and a virtual bird-view camera placed in a fixed position in front of the robot. The virtual images obtained by applying this transformation to the raw images are then passed as input to the GMM. This network outputs the parameters (means and covariance matrices) of a Gaussian Mixture Model (GMM) with 5 components encoding the desired 2D trajectory of the robot hand over the table. Using this model a 2D trajectory over time can be generated by Gaussian mixture regression (GMR): this will

[1]The dataset will be released on the VisLab webpage: `http://vislab.isr.ist.utl.pt/datasets/`

be fed to the robot inverse kinematics solver to generate the final movement of the arm. The system is depicted in Fig.2.

A. Virtual camera

A plane-projective transformation (homography) is applied to the right camera images in order to map those images to a virtual camera placed in a fixed position in front of the robot, at a fixed height from the table. The trasformation between the robot reference frame O and the virtual camera is known precisely. Fig.3 shows coordinate frames involved in this realignment of views to a canonical position (virtual camera, robot and table). The virtual camera points downward to a 1 by 4/3 meters rectangle in the table plane Π. To generate the virtual camera images from the robot right camera we define a homography matrix $\mathbf{H}$ such as:

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

where $\mathbf{X}_r = (x_r, y_r)$ and $\mathbf{X}_v = (x_v, y_v)$ are pixel co-ordinates in the right camera frame and virtual camera frame respectively and z is an arbitrary, non-zero scale factor. This transformation assumes that all the points in the images are co-planar and located at the same height h_t with respect to the table. To calculate the parameters of $\mathbf{H}$, we use equation 1 and the coordinates of four 3D points $(\mathbf{X}(i) = (x(i), y(i), h_t), i = 1, \ldots, 4)$ lying on the table plane Π and their projections on the iCub right camera frame $(\mathbf{X}_r(i) = (x_r(i), y_r(i)))$ and on the virtual camera frame $(\mathbf{X}_v(i) = (x_v(i), y_v(i)))$ respectively. The projections on the right camera frame $\mathbf{X}_r(i)$ are obtained using the iCub forward kinematics and the camera intrinsic parameters K through the function:

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

where $\tau_O^{eye}(q)$ denotes the Denavit-Hartenberg matrix from the robot reference frame O to the right 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 $\mathbf{X}_v(i)$ we use the following function relating the robot reference frame O and the virtual camera image frame:

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

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.

Transformation matrix $\mathbf{H}$ is correct only assuming perfect sensory measurements, but on the real robot we have several sources of errors. Specifically we have:

- Camera calibration errors (small errors in the intrinsic matrix K)
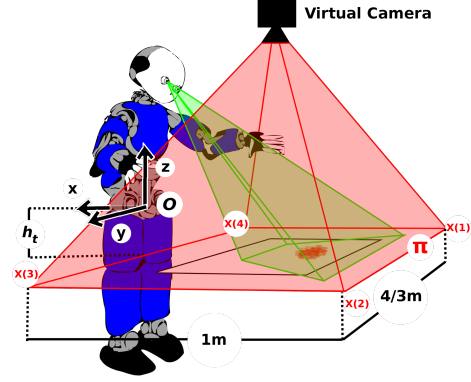- Forward kinematic errors (differences between the kinematics model and the real robot)

Fig. 3. The workspace used in our experiments. The iCub robot is placed in front of a table at an height of h_t from the robot reference frame O. A bird-view virtual camera is placed on top of the table.

Camera calibration errors are independent from the robot pose and joint configurations. Errors on the forward kinematics are composed by a variable term, dependent on joint positions q, and a constant term, the same for every robot configuration. Assuming that the variable term is small enough to be neglected, we focus on the correction of constant errors. Camera calibration and constant kinematics errors can be corrected applying a second transformation matrix $\mathbf{G}$:

$$z \begin{bmatrix} x_v' \\ y_v' \\ 1 \end{bmatrix} = \mathbf{G}\mathbf{H} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} \quad (4)$$

where $\mathbf{X}_v'(i) = (x_v'(i), y_v'(i))$ are the pixel coordinates of the corrected virtual camera images. To calculate $\mathbf{G}$, 4 markers are placed on the table in specific 3D positions $(\mathbf{X}'(i))$, measured placing the robot right hand on top of them and reading the kinematic solver solution. Their pixel coordinates on the right camera image $(\mathbf{X}_v(i) = (x_v(i), y_v(i)))$ are extracted and their expected position in the corrected virtual camera image $(\mathbf{X}_v'(i))$ are calculated using equation 3. $\mathbf{G}$ matrix is calculated only once because errors taken into account are constant and independent from the joint positions q. In Fig. 5 it is possible to see some example of virtual camera images obtained using equation 4. An alternative approach to correct all the kinematic errors (both variable and constant) is to perform a markerless visual servoing, correcting online the hand pose using camera images [17].

B. Convolutional neural network

Once the virtual camera image is created, it is passed as input to the CNN to predict the cleaning trajectory to be performed by the robot's hand. The output of the network are the parameters (mean vectors $\boldsymbol{\mu}$ and covariance matrices Σ) of a 5 components GMM that represents the cleaning trajectory (see subsection II-C for more details). To ensure that the network returns covariance matrices Σ positive-semidefinite and symmetric, we represent Σ using the Cholesky decomposition $\Sigma = LL^*$, where L is a lower

triangular matrix with real and positive diagonal entries, and L^* denotes the conjugate transpose of L.

Since each mixture component is a 3-dimensional Gaussian (xy coordinates in the table plane and time), the network has a total of 45 outputs: the first 15 correspond to the mixture components means, while the last 30 are the lower triangular and diagonal values of L for each mixture component. The output neurons corresponding to the diagonal values of L are forced positive using their exponential instead of the real values.

The network architecture is divided in two parts (see Fig.2): a convolution part based on AlexNet architecture [13] to perform a feature extraction from images, and two fully connected layer to predict the GMM parameters from the features latent layer. The 320x240 virtual camera images pass through 5 convolution layers that generate a latent layer with 4096 neurons. The latent layer is then fully connected to the 45 output neurons.

C. Gaussian mixture model and Gaussian mixture regression

Cleaning trajectories are represented as a time indexed 2-dimensional vector:

$$\mathbf{T}(n) = (n, x(n), y(n)), \qquad n = 0, \ldots, N-1, \quad (5)$$

where N is the number of points that form the trajectory, $\mathbf{T}_n(n) = n$ is a time variable and $\mathbf{T}_{xy}(n) = (x(n), y(n))$ are the coordinates on the table plane Π. Using a GMM to encode a trajectory allows taking into account the variability of the human demonstrations used to train the network [18]. The GMM probability density function is given by

$$\boldsymbol{p}(\mathbf{T}(n)) = \sum_{k=1}^{K} p(k) p(\mathbf{T}(n)|k) \quad (6)$$

where $p(k) = 1/K$ is a uniform prior and $p(\mathbf{T}(n)|k) = \mathcal{N}(\mathbf{T}(n); \boldsymbol{\mu}(k), \Sigma(k))$ is the Gaussian probability density function for $\mathbf{T}(n)$ given the kth mixture component. To generate a trajectory the density function in Eq. 5 is conditioned on time n: this is known as Gaussian Mixture Regression (GMR), resulting in a conditional mean $\hat{\mathbf{T}}_{xy}$ and a conditional covariance $\hat{\mathbf{T}}_{\Sigma,xy}$ for each point n in the trajectory.

D. Loss Function

The loss function minimized during training is:

$$\mathcal{L} = \sum_{n=0}^{N-1} e(n)^T \hat{\mathbf{T}}_{\Sigma,xy}^{-1}(n) e(n) + \gamma \frac{\sum\limits_{n=0}^{N-1} (\hat{\mathbf{T}}_{\Sigma,xy})^2(n)}{N}, \quad (7)$$

with $e(n) = (\mathbf{T}_{xy}(n) - \hat{\mathbf{T}}_{xy}(n))$. The sum at the right side of equation 7 is divided in 2 terms. The first term is the squared error between the spatial component of the kinesthetic trajectory $\mathbf{T}_{xy}$ and its GMR expectation $\hat{\mathbf{T}}_{xy}$, weighted by the inverse of the conditional covariance matrices of each trajectory point $\hat{\mathbf{T}}_{\Sigma,xy}$. The conditional covariance matrices $\hat{\mathbf{T}}_{\Sigma,xy}$ can be seen as a confidence measure of the conditional expectation $\hat{\mathbf{T}}_{xy}$. Weighting the error e with the inverse of
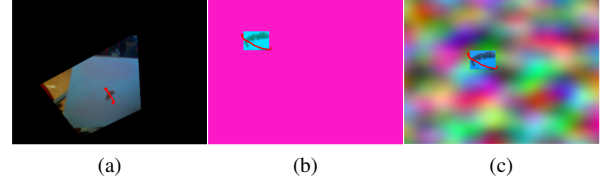

Fig. 4. Elements extracted from the dataset. The trajectories are plotted on top of the images in green. (a): original dataset element (b): shifted, different illumination and uniform color background element (c): shifted, different illumination and perlin noise background element.

$\hat{\mathbf{T}}_{\Sigma,xy}$ gives more importance to the expectations on which we are more confident. Minimizing only the first term of the loss function can result in a growth of the elements of $\hat{\mathbf{T}}_{\Sigma,xy}$ without reducing the actual error e. To solve this problem we introduced the second term: the mean of the squared sum of the elements of each covariance $\hat{\mathbf{T}}_{\Sigma,xy}(n)$ multiplied by a gain factor γ.

III. EXPERIMENTAL SETUP

In this section, we will explain the experimental setup used on the preparation of this manuscript. We introduce the robotic platform used (the iCub), explain the procedure to collect the dataset, how is the system initialized and the evaluation metrics used during tests on a robotic platform.

A. iCub description

The iCub humanoid robot [19] was developed in the context of the EU project RobotCub (2004-2010) and subsequently adopted by more than 35 laboratories worldwide. It has 53 motors that move the hands, arms, head, waist and legs and it has the average size of a 3-year-old child. Its degrees of freedom and human-like appearance are important characteristics that enable the study of human-robot interaction and autonomy in humanoid robots. The stereo vision system (cameras in the eyeballs), proprioception (motor encoders and torque sensors), touch (tactile fingertips and artificial skin) and vestibular sensing (IMU on top of the head) are major features that allow the implementation of the proposed methodology.

B. Collecting the Dataset

We created a dataset consisting on virtual camera images of a dirty table and robot 2D hand trajectories performed to clean it. To collect the dataset we placed a table with size 50x50 cm in front of the iCub at an height h_t of -1 cm from the iCub reference frame O. We recorded 659 trials during which we placed a dirt spot on the table (marker scribbles or clusters of lentils) and guided the robot right hand in order to clean the spot. During the trials a sponge was fixed to the hand of the robot. The robot arm was running using a zero torque controller[2] that permitted to easily move it by hand. The humans were instructed to perform a different

[2]We used the "Force Control" iCub module. The code can be found at https://github.com/robotology/icub-basic-demos/tree/master/demoForceControl

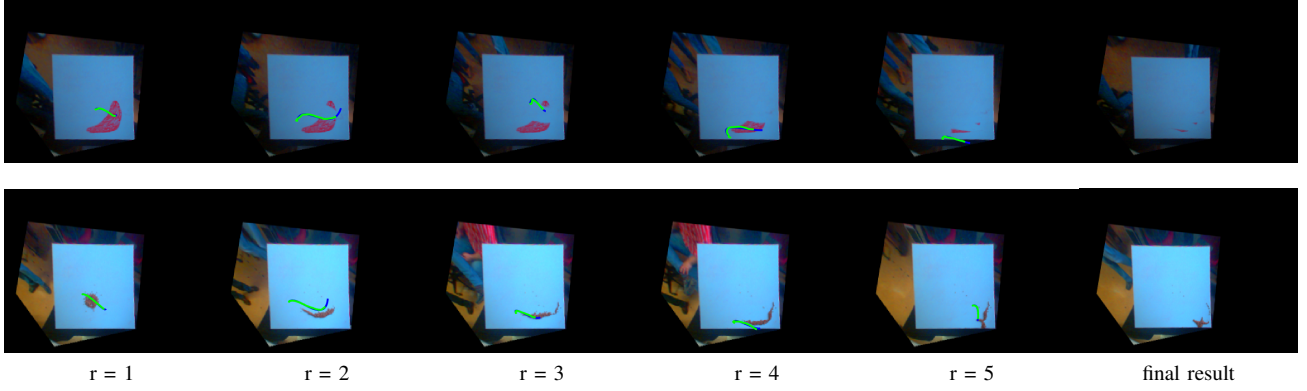| r = 1 | r = 2 | r = 3 | r = 4 | r = 5 | final result |

Fig. 5. Testing examples on the real robot. In blue it is possible to see the output trajectory and in green the trajectory performed by the robot (according with its forward kinematics). Left Column: markers scribbles; Right Column: lentils.

cleaning movement for different kind of dirt: pass on top of marker scribbles and drag to the bottom-right corner of the table the lentils. During the beginning of each trial, the virtual camera image was constructed and the hand trajectory was stored. The hand trajectories were re-sized to 200 points each. During the recording the torso of the iCub was fixed (due to limitation of the zero torque controller) resulting in a limited cleaning area. The dataset was augmented in order to make it more robust to real world scenarios. First the position of both images and trajectories was randomly shifted to augment the cleaning area, second the illumination intensity and color was artificially modified, and third the background around the dirty spots was substituted with random perlin noise or a random uniform color. These changes resulted in a dataset 21 times bigger than the original (13839 elements). Fig. 4 shows an example of 3 images from the dataset.
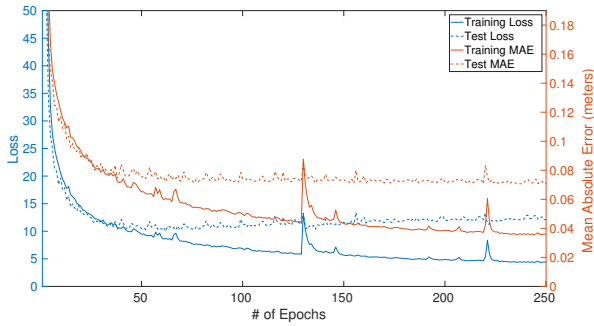

Fig. 6. Loss value evolution during 250 epochs for training and test (blue solid and dashed lines, left y axis). Mean absolute Error (MAE) evolution during 250 epochs for training and test (orange solid and dashed lines, right y axis).

C. System initialization

To test the system we use a scenario similar to the one depicted in Fig. 3. The 50x50 cm table was placed in front of the robot at 3 different heights: -5 cm, -10 cm and -15 cm. In order to demonstrate the robustness of our system to different conditions, we performed the tests using the iCub left hand. The homography matrix for the virtual camera was re-calibrated using the left arm due to different errors

in left and right kinematics chains. For each table height we predefined the z position of the hand during the cleaning movement. The network was trained using 70% of the dataset as training set and the remaining 30% as test set. We stopped the training process after 250 epochs. We used the Adam optimizer to minimize the loss function. The batch size during training was of 200 elements. In order to prevent overfitting, we applied a dropout with ratio 0.5 to the first two fully connected layers of the networks. The networks were implemented using PyTorch[3]. The loss function gain value γ was set equal to 100.

D. Error metrics

To test the system we placed a dirt spot (marker or lentils) on the table and let the robot clean it in 5 repetitions without human intervention. Two different metrics were defined to evaluate the performances of the system. In case of marker scribbles, the percentage of dirty area $m_1(r)$ after each repetition was calculated:

$$m_1(r) = \frac{A(r)}{A(1)}100, \qquad r = 1, \ldots, N_r \qquad (8)$$

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

For the lentils example, the performance metric is the distance $m_2(r)$ between the centroid of the cluster and the bottom-right corner of the table:

$$m_2(r) = \sqrt{(c(r) - o)^T (c(r) - o)} \qquad (9)$$

where $c(r)$ is the centroid of the cluster at repetition r and o is the bottom-right corner of the table both expressed in the robot reference frame (meters).

To evaluate the dirty area and lentils centroids in the images, we used a color (RGB)-based segmentation.

IV. RESULTS

A. Test set results

Fig. 6 shows the performance of the network on training and test set during 250 epochs. On the left y axis, in blue,
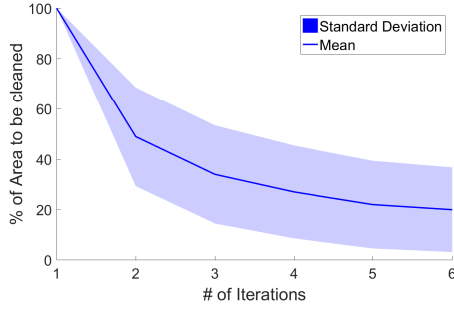
[3]http://pytorch.org

Fig. 7. Mean and standard deviation for 15 Marker Experiments with 3 different table height.
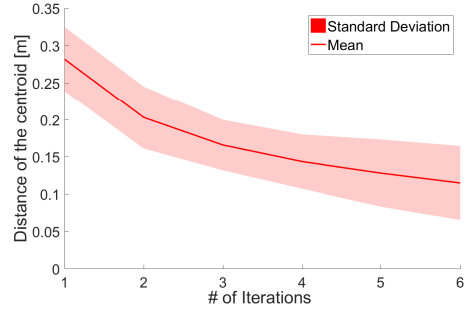

Fig. 9. Mean and standard deviation for 15 Lentils Experiments with 3 different table height
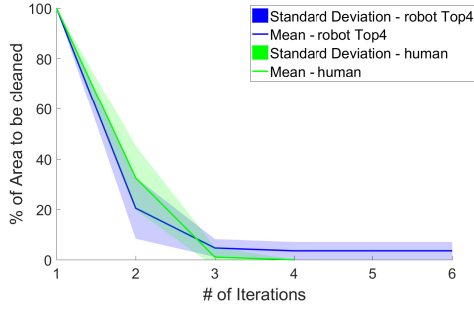

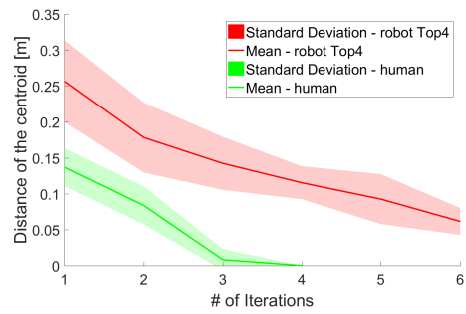Fig. 8. Comparison between the Top-4 marker experiments with the human demonstrations

Fig. 10. Comparison between the top-4 Lentils experiments with the human demonstrations

is represented the loss function $\mathcal{L}$, while on the right y axis, in orange, is represented the Mean Absolute Error (MAE) between the expected trajectory $\hat{\mathbf{T}}_{xy}$ and the kinesthetic demonstration $\mathbf{T}_{xy}$. The figure shows how minimizing the loss function $\mathcal{L}$ results in minimizing the MAE between $\hat{\mathbf{T}}_{xy}$ and $\mathbf{T}_{xy}$. The MAE on the training set after 250 epochs converge to 3 cm while the test MAE converge to 7 cm. The higher error on the test set is due to the high variance in the human demonstrations, making impossible to replicate the exact trajectory position. Nonetheless the network is able to extract general features like dirt position and trajectory direction. Fig. 5 shows how the network is able to create successful trajectories from unseen scenarios.

B. Robot experiments

The proposed architecture was tested on a real scenario using the iCub left arm[4]. In Fig. 5, one can see two trials on the real robot over 5 repetitions without human intervention. In the top row one can find the marker scribbles example, and in the bottom row the lentils. The output trajectory ($\hat{\mathbf{T}}_{xy}$) from the network is in blue and the trajectory performed by the robot according with its forward kinematics is in green . As can be seen in the bottom row of Fig. 5 example r = 3, although the trajectory is on top of the lentils, the robot could not move them in the desired direction, which can be correlated with kinematic errors present in the arm. Indeed, we are learning from demonstration with the right arm and

[4]A video demonstration of the system can be found at the url: `https://youtu.be/RfeIseZO_ng`

adapting the learning to the left one and they have distinct kinematic errors. Nevertheless, as explained in Section III, we are performing a kinematic correction from the right arm to the left which is sufficient in most of the cases.

In a more quantitative analysis, and using the error metric defined on Section III-D., we perform 15 experiments on marker scribbles setting the table at 3 different heights, and the results over the 5 repetitions can be seen in Fig. 7. We reduced the dirt in almost 80% of its initial area with a standard deviation of 15%. Comparing, in Fig. 8, the top-4 experiments performed by the robot (*i.e*, those achieving the lower percentage of area to be cleaned) with the kinesthetic teaching examples, it is possible to see that the robot is cleaning with a similar behaviour. Indeed, it is learning the cleaning motion demonstrated by the human.

In the lentils case, the mean error and standard deviation of 15 experiments with the table at 3 different heights can be seen in Fig. 9. The distance from the bottom right corner of the table (the target point when cleaning lentils) was reduced until 0.115 m with a standard deviation of 0.05 m. The top-4 examples (*i.e*, those achieving the smallest distance from the target point) were compared in Fig. 10 with the demonstration examples performed during the kinesthetic teaching. One can see a small difference between the initial center of mass of the lentils cluster in the two sets. Indeed, and since the robot torso was fixed during the human demonstrations, the arm-reachable space was shorter in the former case than in the robot normal operation. However, the slope of both plots are similar, and we conjecture that the robot is learning
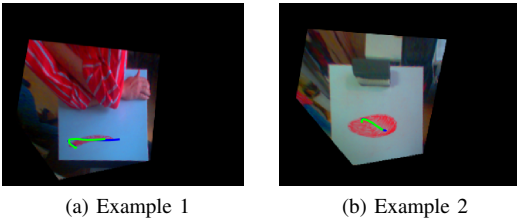
(a) Example 1   (b) Example 2

Fig. 11. Showing Robustness of the algorithm to external objects. (a) a human with a red shirt; (b) a whiteboard eraser.

a cleaning motion resembling the ones demonstrated while performing them on a wider space, achieving the final goal with a few iterations more. Regarding the robustness of the algorithm, we incorporate external object on the scene. In Fig. 11, a human with a red shirt (a) and a whiteboard eraser (b) were on the top of the table. The network was able to ignore them and to produce a trajectory (in blue) which was performed by the robot (in green) being able to clean the table from a marker scribble.

V. CONCLUSIONS AND FUTURE WORK

We presented a novel approach for autonomous robotic cleaning tasks, based on a deep learning architecture, that learns how to generate 2D cleaning trajectories over a table from a set of human kinesthetic demonstrations. Data augmentation techniques have been used to increase the robustness of the network to disturbances and environmental changes. A plane-projective transformation was successfully used to make the system invariant to the robot posture. We have shown that, after training the CNN, the robot is able to successfully generate cleaning trajectories with its arm. We also show that the knowledge acquired from the kinesthetic demonstrations using the right arm can be straightforwardly transferred to the left one. Experiments with the iCub were presented and discussed. We argue that our work can be easily extended to different types of cleaning motion describable by 2D hand trajectories (*e.g.*, spread a cleaning product on a table). New cleaning behaviours can be learned by the system simply adding new kinesthetic demonstration to the training set. In future work, we intend to expand the number of cleaning movements and further investigate if such learned skills can be transferred to other types of robots.

ACKNOWLEDGMENTS

REFERENCES

[1] F. Cavallo, R. Limosani, A. Manzi, M. Bonaccorsi, R. Esposito, M. Di Rocco, F. Pecora, G. Teti, A. Saffiotti, and P. Dario, "Development of a socially believable multi-robot solution from town to home," *Cognitive Computation*, vol. 6, no. 4, pp. 954–967, 2014.

[2] D. McColl, A. Hong, N. Hatakeyama, G. Nejat, and B. Benhabib, "A survey of autonomous human affect detection methods for social robots engaged in natural hri," *Journal of Intelligent & Robotic Systems*, vol. 82, no. 1, pp. 101–133, 2016.

[3] G. Ferri, A. Manzi, P. Salvini, B. Mazzolai, C. Laschi, and P. Dario, "Dustcart, an autonomous robot for door-to-door garbage collection: From dustbot project to the experimentation in the small town of peccioli," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 655–660.

[4] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba, "Humanoid motion generation system on hrp2-jsk for daily life environment," in *Mechatronics and Automation, 2005 IEEE International Conference*, vol. 4. IEEE, 2005, pp. 1772–1777.

[5] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," in *International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 107–114.

[6] J. Hess, G. D. Tipaldi, and W. Burgard, "Null space optimization for effective coverage of 3d surfaces using redundant manipulators," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 1923–1928.

[7] D. Leidner, W. Bejjani, A. Albu-Schäffer, and M. Beetz, "Robotic agents representing, reasoning, and executing wiping tasks for daily household chores," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1006–1014.

[8] F. Cruz, S. Magg, C. Weber, and S. Wermter, "Training agents with interactive reinforcement learning and contextual affordances," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 4, pp. 271–284, 2016.

[9] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3970–3975.

[10] S. Calinon, T. Alizadeh, and D. G. Caldwell, "On improving the extrapolation capability of task-parameterized movement models," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 610–616.

[11] T. Alizadeh, S. Calinon, and D. G. Caldwell, "Learning from demonstrations with partially observable task parameters," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3309–3314.

[12] J. Kim, N. Cauli, P. Vicente, B. Damas, F. Cavallo, and J. Santos-Victor, ""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)*, 2018.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[15] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: a survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[16] U. Pattacini, "Modular cartesian controllers for humanoid robots: Design and implementation on the icub," Ph.D. dissertation, Italian Institute of Technology, 2011.

[17] P. Vicente, L. Jamone, and A. Bernardino, "Towards markerless visual servoing of grasping tasks for humanoid robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3811–3816.

[18] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[19] G. Metta and L. Natale and F. Nori and G. Sandini and D. Vernon and L. Fadiga and C. von Hofsten and K. Rosander and M. Lopes and J. Santos-Victor and A. Bernardino and L. Montesano, "The iCub humanoid robot: an open-systems platform for research in cognitive development," *Neural Networks*, vol. 23, 2010.