Efficient Robotic Grasp Learning by Demonstration

Ziyan Gao and Nak Young Chong

Japan Advanced Institude of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan
{ziyan_gao2015, nakyoung}@jaist.ac.jp

Abstract. In this paper, we propose a Learning from Demonstration approach for robotic grasping with compliant arms. The compliance in the robot arm for safety often causes a problem in grasping. In our approach, we construct a recurrent neural network, given the estimation of the target object position and random initial joint angles of the robot arm, that can produce the whole trajectories for grasping the target object. In order to generate smooth and stable trajectories and to decrease the number of human demonstrations, we propose a data augmentation method to increase the training data and utilize the trajectory planning technique using cubic splines for smooth and stable trajectories. Specifically, the two arms of the robot are trained respectively, and a support vector machine is used to decide which arm needs to be used for grasping the target object. The evaluation results show that our recurrent model not only has a good prediction for the final joint configurations, but also generates smooth and stable trajectory. Moreover, the model is robust to the changes in the initial joint state which means that even though the initial joint configuration is affected by disturbances, the model can still generate trajectories leading to the final joint configurations for grasping the object. Finally, we tested the proposed learning method on the Pepper robot which can successfully grasp randomly placed object on the workbench. Compared to traditional methods which need to avoid singular configurations as well as to secure accurate localization, our method turns out to be robust and efficient and can be applied to cluttered environment.

Keywords: grasp planning, Learning from Demonstration, recurrent neural network, support vector machine

1 Introduction

Recent advances in robotic grasping have shown promising results. However, to make robots see, perceive, decide, and act in a way a human or a primate does, many challenges still need to be addressed [1]. In recent years, Learning from Demonstration (LfD) was successfully used in the field of robotics for applications such as playing table tennis [2], object manipulation [3], making coffee

[4], grasping novel objects [5], carrot grating [6], etc. Since robots must operate in real environments and make decisions based on noisy sensory information and incomplete models of the environment, deep learning methods that directly model the relationship between the available sensory input and the desired output have become more popular [7]. In order to generate smooth trajectories and to decrease the number of human demonstrations, we propose a data augmentation method to increase the training data and utilize the trajectory planning technique for smooth and stable trajectories. For human-like dual-arm robots, it also needs to make decision for which arm needs to be used for grasping the object. We implemented a support vector machine classifier for the arm selection problem.

2 Ralated Work

A major challenge in LfD is to extend these demonstrations to unseen situations [8]. One obvious way to mitigate this problem is by acquiring a large number of demonstrations covering as many situations as possible [9]. Some researchers proposed cloud based and crowdsourced data collection techniques [10],[11],[12] or the use of simulation environments [13]. Another direction is to use smaller number of demonstrations, and update the learning model for better generalization. One possible technique is to hand-engineer task-specific features [14],[15]. [16] uses a large amount of synthesized images for training a model for position detection and transfers to the real physical environment images using a handful of images collected in the real physical world. Our method, in contrast to the previous approach, augments the data based on the demonstrated data. [17] uses a recurrent model to pick and place an object in a virtual environment and deals with the pick and place task both by recurrent neural network (RNN) and reinforcement learning. [18] uses a deep spatial autoencoder to acquire a set of feature points that describe the environment for the task. In our approach, we estimate the location only by the robot head orientation and object loaction in an RGB image.

3 Our Approach

3.1 An Overview of Our Approach

An overview of our approach is illustrated in Fig. 1. There are four phases: Data Collection Phase, Trejectory Generation Phase, SVM Training Phase, and Trajectory Generator Training Phase. In the Data Collection Phase, we collect multiple sets which can be represented as:

$$\{C_x, C_y, H_p, H_y, J\},$$

where C_x, C_y refer to the coordinates of the location of the object in the image plane, H_p, H_y refer to the neck joint angles of the robot, and J refers to the

joint angles of left or right arm. We use J_0 to represent the initial joint angles and J_T to represent the final joint angles. In the Trajectory Generation Phase, we use cubic polynomial to generate the whole trajectory and use our data augmentation method to create multiple trajectories based on the collected data. In the Trajectory Generator Training Phase, we use the augmented data to train the recurrent neural network. In the SVM Training Phase, we use the $\{C_x, C_y, H_p, H_y\}$ as input and binary signal (0 represents the left arm and 1 represents the right arm) to train a support vector machine with non-linear kernel.

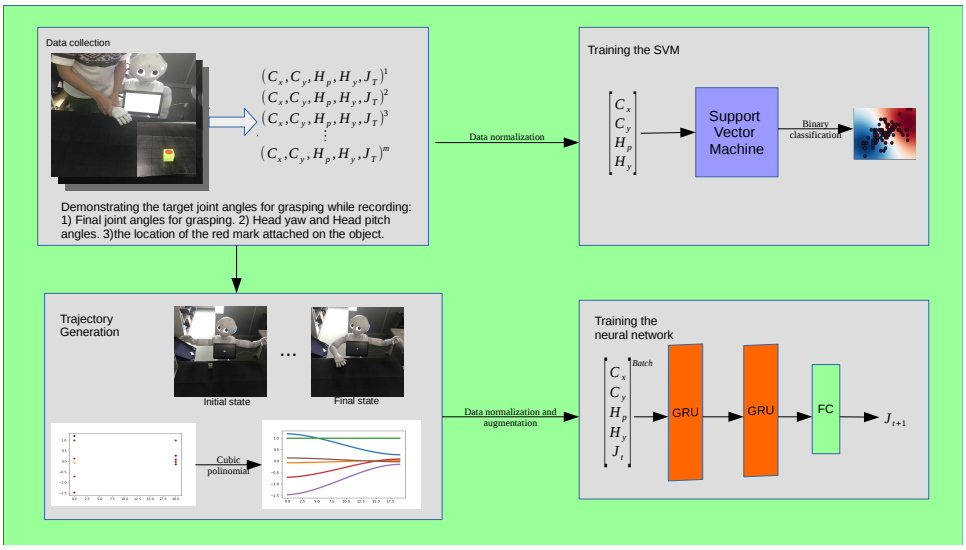


Fig. 1. An overview of our approach.

3.2 Trajectory Planning in Joint Space

We used a cubic polynomial represented as

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{1}$$

to generate the trajectory connecting the initial joint angle and final joint angle of the robot arm. $\theta(t)$ is the joint angle function about time t. Four constraints must be specified to solve the unknowns: $\{a0, a1, a2, a3\}$. The first two constraints are the start and end configurations, and the last two are the initial and end velocities. t_0 identifies the initial time and t_f identifies the final time. The

constraints can be represented as

$$\begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & t_f & 2t_f & 3t_f^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \theta(t_0) \\ \dot{\theta}(t_0) \\ \theta(t_f) \\ \dot{\theta}(t_f) \end{pmatrix} \tag{2}$$

T represents the coefficient matrix. Let A represent the unknowns, and θ represent the functions, respectively. Then the unknowns can be derived as:

$$A = T^{-1}\theta \tag{3}$$

Now we can collect the data without noisy signals. This will be conductive to training the recurrent model much more efficiently. After obtaining the cubic polynomial, we sampled 21 trajectory points with a uniform interval from the cubic polynomial (see Fig. 2).

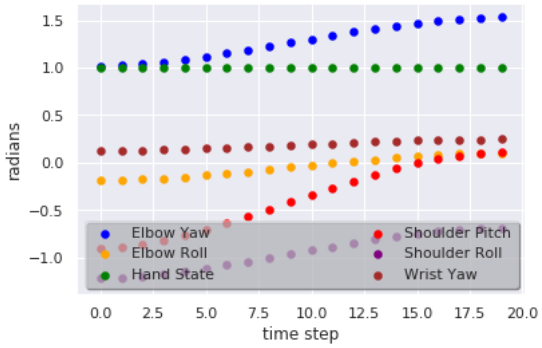


Fig. 2. We sampled 21 sampling points from each of the cubic polynomial: the first sampling point of each polynomial will be the input of the recurrent neural network, and the rest of the sampling points will be the output of the recurrent neural network.

3.3 Data Augmentation

In order to increase the training data, we generate multiple initial states in a small vicinity of the initial joint state. We generated 243 initial states both for the right and left arms. After that we used cubic polynomial to connect these initial states and final states. Therefore, we can generate 243 trajectories for one $\{C_x, C_y, H_p, H_y, J_T\}$ and each of them has 21 sampled via points. There are two merits in our method: First, our model can be trained by much larger data so that it will predict much more smooth and stable trajectories. Secondly, our

model is non-sensitive to the initial joint state, which means that even though the initial state of the robot is affected by disturbances, it does not cause an error in the model's prediction of the final joint state.

3.4 Trajectory Generator and SVM

We use a recurrent neural network as the trajectory generator. Basically our model is an one to many model, which consists essentially of (1) two Gated Recurrent Unit layers [19] and (2) one fully connected layer. The robot controller takes $\{C_x, C_y, H_p, H_y, J_0\}$ as input and outputs the whole trajectories leading to the final joint angle. There are two ways to train the recurrent model as shown in Fig. 3. The first one is trained by a sequence to sequence fashion. The input is the first 20 trajectory points, the output is the last 20 trajectory points. In other words, the output shifts backward by one time step compared to the input. The second one is trained in a one to many fashion. The input is the first trajectory point, the output are the last 20 trajectory points. Once fitting the first trajectory point into the model, then the model needs to generate the whole trajectory. During the training phase, the first one converged faster than the second one and the loss function also decreased to less than 10^{-6}. But in the test phase, the model trained in the first fashion tends to result in nonsuitable trajectories, while the model trained in the second fashion can generate smooth as well as accurate trajectories even though it is difficult to converge. Furthermore, we implemented a Support Vector Machine classifier for hand selection. The input features are selected as $\{C_x, C_y, H_p, H_y\}$, and the output is a binary signal which inferred to use the left or right arm. We use Radial Basis Function kernel SVM given by

$$K\left(x^{(i)}, x^{(j)}\right) = \phi(x^{(i)})^T \phi(x^{(j)}) = \exp(-\gamma \|x^{(i)} - x^{(j)}\|^2) \tag{4}$$

where γ defines how far the influence of a single training example reaches. The data flow is shown in Fig. 4.

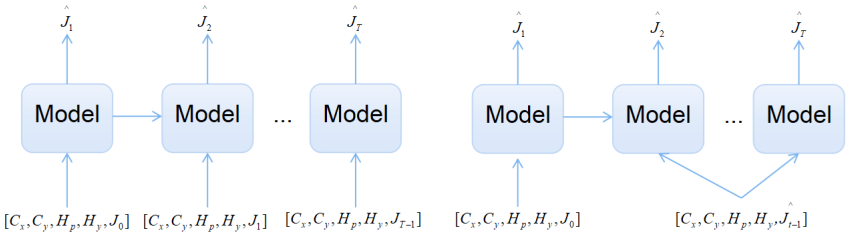


Fig. 3. Different training approach for trajectory genteration model.

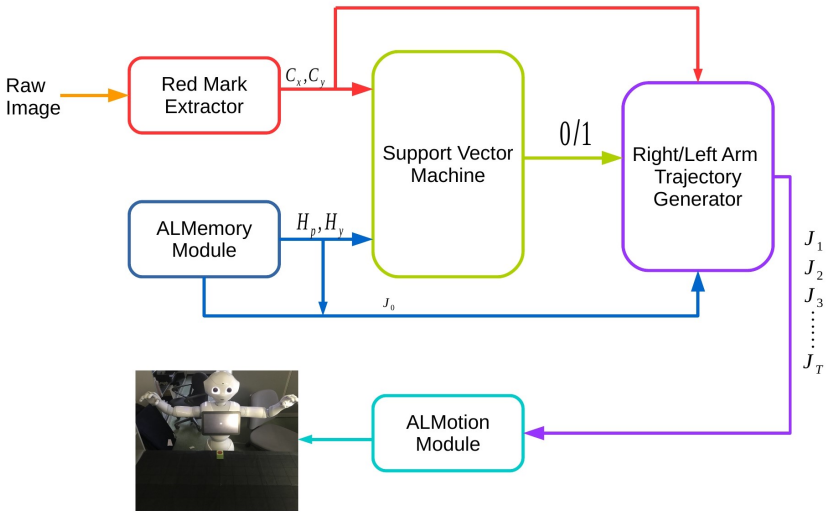


Fig. 4. A schematic of the proposed robot grasping learning. Raw image is an RGB image of size 240×320, c_x, c_y are the coordinates of the center of the object related to the top left corner of the image. H_p, H_y represent two neck angles: head pitch and head yaw. $J_1, J_2, ..., J_T$ represent the joint angles of left/right arm in the corresponded time step.

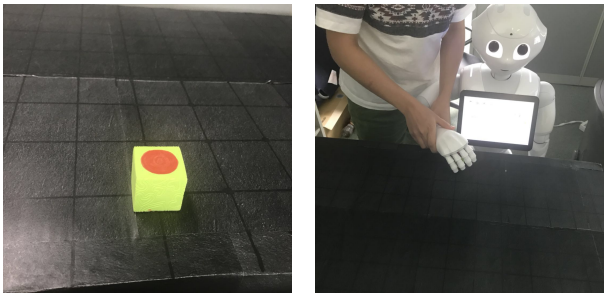
4 Experiment

4.1 Physical Environment

In order to simplify the task for object localization, we made a black colored workbench whose height is 83 centimeters from the floor. We used a 3.5 centimeter cubic block (see Fig. 5(a)) and attached a red mark on top of the surface of the block to be the target. For convenience, we marked 24 positions on the workbench with the same intervals of 5 centimeters. In the experiment, the Pepper robot (see Fig. 5(b)) is used to collect the data as well as to test the proposed recurrent model. The RGB camera mounted on the mouth of the Pepper robot is used for recording the instant picture. The instant joint angles of the robot's left or right arm are also recorded. During the phase of collecting data, we fixed the robot's position and the waist as well as knee joint angles. We place the block on the workbench, and then the robot grabs pictures with different head orientations. After finishing taking picture and recording the head orientation, we guided the robot's arm to the desired position and recorded the joint angles of robot's right or left arm.

4.2 Data Preprocessing

In the experiment, we collect 2 demonstrated initial joint angles, 24 demonstrated approaching joint angles for the robot's right or left arm in total, and



(a) cubic block (b) human demonstration

Fig. 5. the experimental setup

574 images and their corresponding head orientations. After that we generate 139,482 trajectories in total. Finally, we normalized all the sample points to the range between 0 and 1.

4.3 Training for the recurrent model and SVM

We set 40 neurons in each hidden recurrent layer and 6 neurons for the output layer, and did not add any activations on the output layer. In the training phase, we set the learning rate to be 0.001 and use Adam optimizer to train the model. After 2,000 iterations, we stop the iteration process.

We developed a Support Vector Machine for hand decision. There are two hyper-parameters we need to set: C and γ, The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane mis-classifies more points. γ can be seen as the inverse of the radius of the influence of samples selected by the model as support vectors.

5 Result

For the evaluation of the recurrent model for trajectroy generation, first we use the trained recurrent model to predict the joint angles on the test set. Then we use the forward kinematics to obtain the position of end effector relative to the robot torso. The Pepper robot has 5 degrees of freedom for each arm, and we compute the position of cubic block with respect to the robot torso.

Using the transformation function, the end effector's position as well as orientation can be calculated. We only evaluate the position error relative to the demonstrated position. We use the mean squared error to calculate the distance between the generate predicted hand position and the demonstrated position.

Finally, we visualize the error distribution by using the seaborn library as shown in Fig. 6.

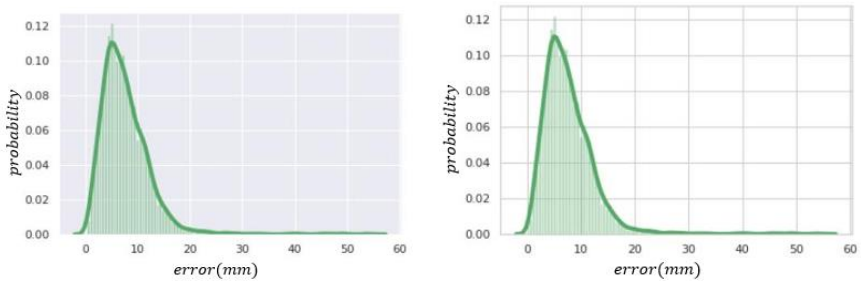


Fig. 6. the mean squared error distribution both of robot hand: The left one represents for left hand position error distribution, its mean and variance are 6.35 and 14.07, respectively. The right graph shows the right hand position error distribution, its mean and variance are 7.07 and 23.74 respectively.

In order to verify that our model is robust to the changes in the initial joint state, we randomly choose the initial joint state in the trajectory generation model. The generated whole trajectory is then compared to the demonstrated trajectory. The result is shown in Fig. 7. It is clear that the robot hand finally reaches to the same position even though the initial state is different. The grasping procedure is shown in Fig.8

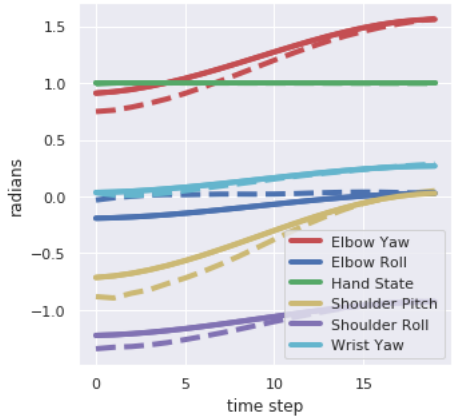


Fig. 7. Different initial joint angle states leading to the same target state: The dash lines represent the trajectories generated by the recurrent model, while the solid lines represent the trajectories demonstrated.

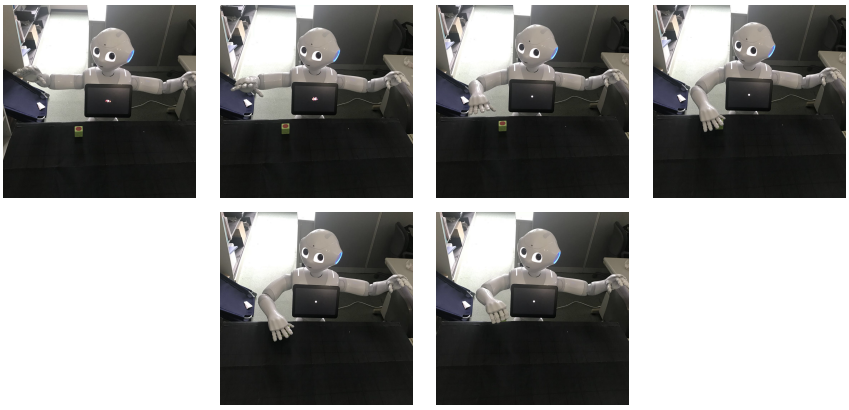


Fig. 8. Pepper robot grasps the cubic block, approaching the final joint states from its initial joint states.

We use $\{C_x, H_p\}$ as the feature for visualization. Fig. 9 shows that when C equals to 100 and γ equals to 0.1, the SVM exhibit the best classification performance on the test set.

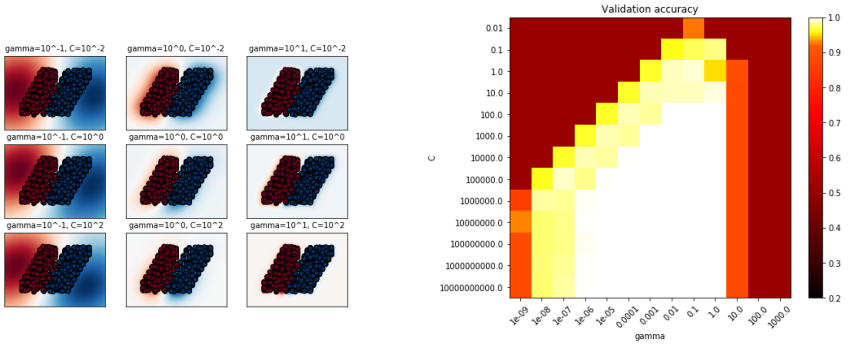


Fig. 9. Visualization of SVM with different hyperparameters.

We compare the models which are trained by different training sets: one is the augmented by the proposed method, the other one is the origin dataset. We also use forward kinematics to compute the hand position relative to the robot

torso and then use mean squared error to calculate the error between the taught position and the calculated position. The result is shown in Fig. 10. It shows that the model trained by augmented data outperforms the model trained by the original demonstrated data.

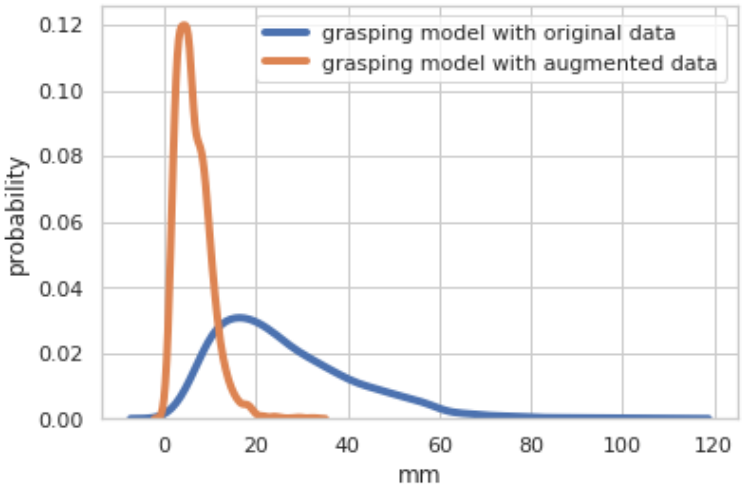


Fig. 10. Performance comparison between the model trained by augmented data and the original data. The model trained by augmented data has a smaller error, which is the distance between the calaulated hand position at the end of the generated trajectory and the demonstrated hand position.

During the data augmentation section, we augmented the data by randomly changing the initial state within a small range, which is 0.2 radians for each of the joint. After finishing training the model, we random selected a test data from the test set, and generated 1,000 initial states by adding a small perturbation to each of the joint values. Then we input these initial states to our model and pick up the final states. We used forward kinematics to calculate the hand position relative to robot torso and compared with the demonstrated position. We use mean squared error to calculate the distance between them. We found that the perturbed initial joint angles can be tolerated up to a maximum of 0.2 radians, and clearly do not lead to the prediction error. Fig. 11 shows the error distribution due to initial perturbations.

6 Conclusion and Future Work

We proposed a new grasp learning by demonstration algorithm for a dual arm humanoid robot with joint compliance. We have learned that the recurrent model can generate stable and smooth trajectories for grasping the object and this

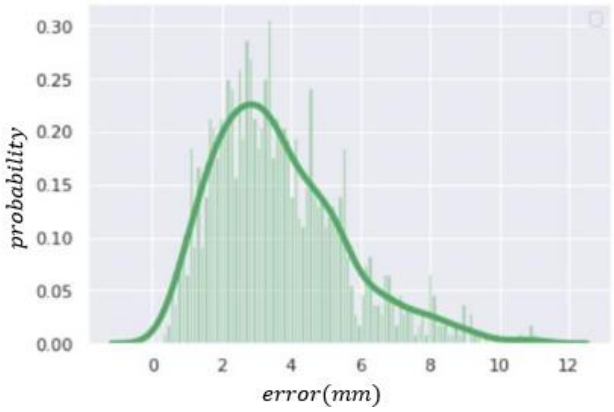


Fig. 11. We change each of the initial joints by adding small perturbations less than 0.2 radians to each of the joints. This graph shows the error distribution, bounded by 12.4 millimeters.

model is robust to the changes in the initial state of robot arm joints. Our proposed data augmentation method was very successful in improving the convergence of the recurrent neural network and the smoothness of the trajectory. The support vector machine classifier with non-linear kernel was capable of deciding which arm needs to be used for grasping based on the head orientation and object location in RGB image features. The proposed model has some limiations: first of all, it cannot accurately generate trajectories when the object is placed on a different height workbench. Inspired by humans, we used the camera twice to see the object with different head orientations, and then used it to train the same model with different input sizes. The result was encouraging, but still needs to be improved. We also tried to use the depth camera mounted on the robot's right eye. However, due to the measurable range limitations, it cannot sometimes detect the object. We will further increase the robot's grasping range and capability by taking its waist joint and mobility and the object's shape into consideration.

Acknowledgment

This project was supported by the EU-Japan coordinated R&D project on "Culture Aware Robots and Environmental Sensor Systems for Elderly Support" commissioned by the Ministry of Internal Affairs and Communications of Japan and EC Horizon 2020.

References

[1] Task-Informed Grasping (TIG) for rigid and deformable object manipulation. https://www.birmingham.ac.uk/research/activity/metallurgy-

materials/robotics/workshops/task-informed-grasping-objects-manipulation.aspx.
[2] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell and A. G. Billard.: Learning and Reproduction of Gestures by Imitation. IEEE Robotics and Automation Magazine, vol. 17, no. 2, pp. 44-54, June 2010.
[3] P. Pastor, H. Hoffmann, T. Asfour and S. Schaal.: Learning and generalization of motor skills by learning from demonstration. 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 763-768. doi: 10.1109/ROBOT.2009.5152385
[4] J.Sung, S.H.Jin, and A.Saxena.: Robobarista: Object part-based transfer of manipulation trajectories from crowd-sourcing in 3d point-clouds. International Symposium on Robotics Research (ISRR),2015.
[5] M.Kopicki, R.Detry, M.Adjigble, R.Stolkin, A.Leonardis, and J.L.Wyatt.: One-shot learning and generation of dexterous grasps for novel objects.The International Journal of Robotics Research.vol.35, no.8,pp.959976,2016.
[6] A. L. P. Ureche, K. Umezawa, Y. Nakamura and A. Billard.: Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations. IEEE Transactions on Robotics, vol. 31, no. 6, pp. 1458-1471, Dec. 2015.doi: 10.1109/TRO.2015.2495003.
[7] Rok Pahic.: Deep learning in robotics.
[8] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning.: A survey of robot learning from demonstration. Robot. Auton. Syst. 57, 5 (May 2009), 469-483. DOI=10.1016/j.robot.2008.10.024. http://dx.doi.org/10.1016/j.robot.2008.10.024.
[9] Rahmatizadeh, Rouhollah and Abolghasemi, Pooya and Blni, Ladislau and Levine, Sergey. (2017). Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-To-End Learning from Demonstration.
[10] . Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, Cloud-based robot grasping with the Google object recognition en- gine, in IEEE International Conference on Robotics and Automation (ICRA) , pp. 42634270, 2013.
[11] M. Forbes, M. J.-Y. Chung, M. Cakmak, and R. P. Rao, Robot programming by demonstration with crowdsourced action fixes, in Second AAAI Conference on Human Computation and Crowdsourcing , 2014.
[12] C. Crick, S. Osentoski, G. Jay, and O. C. Jenkins, Human and robot perception in large-scale learning from demonstration, in Interna- tional conference on Human-robot interaction , pp. 339346, ACM, 2011.
[13] Z. Fang, G. Bartels, and M. Beetz, Learning models for constraint- based motion parameterization from interactive physics-based simula- tion, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) , pp. 40054012, 2016.
[14] 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 , vol. 37, no. 2, pp. 286298, 2007.
[15] S. Calinon, F. Dhalluin, D. G. Caldwell, and A. Billard, Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework., in IEEE International Conference on Humanoid Robots (Humanoids) , pp. 582588, Citeseer, 2009.
[16] Tadanobu Inoue , Subhajit Chaudhury , Giovanni De Magistris and Sakyasingha Dasgupta.: Transfer learning from synthetic to real images using variational autoencoders for robotic applications.(2017)
[17] Giovanni De Magistris, Asim Munawar, Phongtharin Vinayavekhin.: Teaching a Robot Pick and Place Task using Recurrent Neural Network. ViEW2016, Dec 2016, Yokohama, Japan. $< hal - 01426846 >$

[18] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, Pieter Abbeel.: Deep Spatial Autoencoders for Visuomotor Learning. arXiv:1509.06113 (2015)
[19] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555(2014)
[20] http://doc.aldebaran.com/2-4/naoqi/index.html